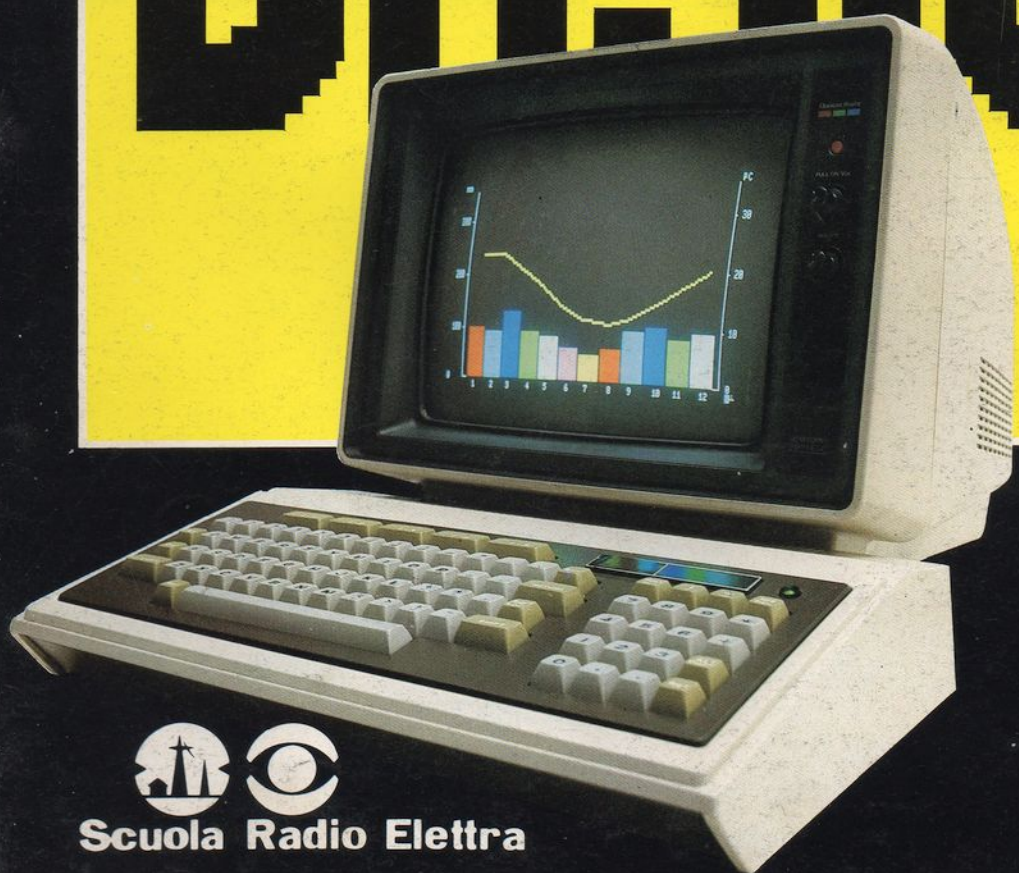
PARLA BASIC



Scuola Radio Elettra

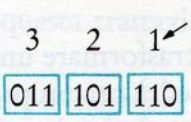
Pagine mancanti

L'aritmetica fondata sui numeri binari è molto importante, perché viene usata per istruire i computer e per rappresentare i dati presenti nei loro circuiti.

Oltre che i sistemi di numerazione visti sopra, i computer utilizzano altri sistemi, che sono agevolmente deducibili quasi tutti dal sistema di numerazione binario.

Essi si ricavano raggruppando le cifre 0 e 1 (bit) in gruppi di tre, oppure quattro e trasformando questi raggruppamenti in valori del nuovo sistema di numerazione.

Per esempio, il numero binario 11101110, equivalente al numero decimale 238, si può trasformare nel **sistema ottale** (ossia, in base 8) raggruppando a tre a tre le cifre binarie a partire da destra:



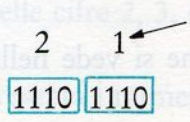
Nel gruppo 1 si ha cosí il numero binario 110, equivalente al numero decimale 6, cioè al numero ottale 6; nel gruppo 2 il numero binario 101, equivalente al numero decimale 5, cioè al numero ottale 5; nel gruppo 3 il numero binario 011 (la cifra 0 è stata anteposta alle cifre 11), equivalente al numero decimale 3, cioè al numero ottale 3.

Il numero ottale corrispondente al numero binario 11101110 è quindi:

3 5 6

Come si è potuto rilevare da questo esempio, quando l'ultimo gruppo di bit a sinistra è costituito da una o due cifre, per continuare ad applicare il metodo di conversione è sufficiente assumere che le cifre mancanti siano costituite da 0 (zero).

Se invece si vuole rappresentare il numero binario 11101110 nel **sistema esadecimale** (sistema di numerazione costituito da quindici simboli, come illustrato nella tabella della *fig. 2*), occorre raggruppare il numero binario **in gruppi di 4 bit**, cioè nel seguente modo:



SISTEMA REALE O DECIMALE	SISTEMA OTTALE	SISTEMA ESADECIMALE
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	10	8
9	11	9
10	12	A
11	13	B
12	14	C
13	15	D
14	16	E
15	17	F
16	20	10

Fig. 2 - I primi sedici numeri interi rappresentati nei sistemi decimale, ottale ed esadecimale.

Nel gruppo 1 si ha cosí il numero binario 1110, equivalente al numero decimale 14, il quale corrisponde (ved. tabella della *fig. 2*) al numero esadecimale E (lettera maiuscola).

Anche il gruppo 2 equivale allo stesso numero esadecimale E.

Quindi il numero esadecimale completo corrispondente al binario 11101110 è:

E E

Analogamente, il numero esadecimale corrispondente al binario 11111001 è:

F 9

infatti 1001 = 9 e 1111 = F; il numero esadecimale corrispondente al binario 10111010 è:

B A

poiché 1010 = A e 1011 = B.

2. - Le quattro operazioni elementari

Le quattro operazioni elementari (somma, sottrazione, moltiplicazione, divisione) sono alla base di ogni tipo di calcolo numerico, anche del piú complicato; si può dire che tutta l'aritmetica è costruita su queste semplicissime operazioni e lo stesso computer è dotato di pochi circuiti elementari aritmetici (somma, sottrazione, moltiplicazione, spostamenti, confronti, complementazione) in grado di sviluppare tutto un complesso di altre operazioni aritmetiche, di calcolare funzioni matematiche e di manipolare e trasferire stringhe di caratteri da zona a zona della memoria.

Il computer, però, per eseguire le operazioni aritmetiche o logiche su numeri, simboli e parole, li trasforma in codici binari, perché la sua logica si basa proprio sulla condizione che le entità di elaborazione siano predisposte e rappresentate in memoria secondo la codifica binaria.

Analogie tra sistema decimale e sistema binario

In questo capitolo si rivedranno regole aritmetiche per l'esecuzione delle quattro operazioni elementari, cercando di stabilire delle analogie operative tra il sistema decimale ed il sistema binario.

a) Addizione tra due numeri decimali positivi

Volendo calcolare la somma di due numeri decimali positivi, prima si incolonnano i numeri sulla virgola (se c'è) e poi si iniziano a sommare le cifre corrispondenti della stessa colonna, procedendo da destra verso sinistra e computando nella somma sempre l'eventuale riporto precedente; ad esempio, per eseguire la somma dei numeri +3,5 e +44,65, prima si incolonnano i numeri +3,5 e +44,65 cosí:

```
  + 3,5
 +44,65
```

poi, a partire dalle cifre decimali piú a destra, si eseguono ripetutamente le addizioni parziali:

```
0+5=[5]          riporto 0
5+6=[1]          riporto 1
3+4=7+1=[8]      riporto 0
0+4=[4]          riporto [0] (l'ultimo riporto non è significativo)
```

cioè si ottiene:

$$\begin{array}{r} +\ 3{,}5\ + \\ +44{,}65= \\ \hline +48{,}15 \end{array}$$

Se si dovessero sommare due numeri decimali interi, ad esempio +48 +69, si procederebbe prima ad un incolonnamento sulle cifre intere (la virgola è implicitamente posta a destra della cifra intera meno significativa):

$$\begin{array}{r} +48 \\ +69 \end{array}$$

e, sempre partendo da destra, si calcolerebbero poi le seguenti cifre del risultato:

$8+9=\boxed{7}$ riporto 1

$4+6=10+1=\boxed{1}$ riporto $\boxed{1}$

cioè:

$$\begin{array}{r} +48+ \\ +69= \\ \hline +117 \end{array}$$

b) Addizione tra due numeri binari positivi

L'addizione tra due numeri binari positivi (cioè maggiori di 0) avviene analogamente a quanto descritto nel punto a) per i numeri espressi nel sistema decimale, tenendo presenti i seguenti casi elementari:

$$\begin{array}{l} 0+0=0 \\ 0+1=1 \\ 1+0=1 \\ 1+1=10 \end{array}$$

Per esempio, la somma dei due numeri binari frazionari sotto riportati si esegue incolonnando dapprima gli stessi numeri sulla virgola e completando le cifre binarie del secondo addendo con due zeri davanti (cioè lo si considera 0011,011)

$$\begin{array}{r} +1101{,}010 \\ +\ \ 11{,}011 \end{array}$$

e poi eseguendo le somme delle cifre binarie corrispondenti, ossia:

0+1= 1	riporto 0
1+1= 0	riporto 1
0+0=0+1= 1	riporto 0
1+1= 0	riporto 1
0+1=1+1= 0	riporto 1
1+0=1+1= 0	riporto 1
1+0=1+1= 0	riporto 1

cioè:

```
  1101,010+
+ 0011,011=
-----------
+10000,101
```

Per sapere a quale numero decimale frazionario corrisponde il risultato appena ottenuto, occorre eseguire le trasformazioni dal sistema binario al sistema decimale precedentemente viste. Analogamente, per sommare i numeri binari interi positivi:

```
+11101
+10001
```

si eseguono le addizioni:

1+1= 0	riporto 1
0+0= 0+1= 1	riporto 0
1+0= 1	riporto 0
1+0= 1	riporto 0
1+1= 0	riporto 1

cioè:

```
  11101+
+ 10001=
--------
+101110
```

c) Addizione e sottrazione tra numeri relativi

I numeri relativi, come è noto dall'aritmetica, sono valori dotati di segno "+" per i numeri positivi (maggiori di 0) e di segno "–" per i numeri negativi (minori di zero).

Nel computer le operazioni di addizione e sottrazione tra due numeri relativi vengono eseguite con una tecnica particolare, detta **tecnica del complementare**, la quale consiste nel trasformare il valore negativo nel complemento alla base e poi nell'eseguire la somma tra i numeri cosí ottenuti.

Alla fine di questa operazione viene risottratta la base, cioè viene di nuovo "complementato" il risultato.

Ecco un esempio fatto con il sistema decimale; si supponga di dover eseguire la sottrazione tra i valori:

$$+37 \text{ e } -42$$

con la tecnica del complementare.

Il numero negativo –42 viene "complementato" al valore 100, cioè viene calcolata la differenza (o somma algebrica):

$$\begin{array}{r} 100 - \\ 42 = \\ \hline 58 \end{array}$$

Il complemento a 100 di –42 è quindi +58.

Quest'ultimo viene sommato a 37:

$$\begin{array}{r} +37 + \\ +58 = \\ \hline +95 \end{array}$$

Infine si sottrae da quest'ultimo la base 100 (che era stata "artificialmente" aggiunta per consentire il calcolo del complemento), ossia si fa l'operazione 95 – 100, differenza che si può anche scrivere:

$$-(100-95)$$

e quindi calcolare cosí:

$$\begin{array}{r} -100- \\ +\ 95= \\ \hline -\ \ 5 \end{array}$$

che è appunto il risultato della somma algebrica tra i numeri relativi +37 e −42.

Perché si è ricorsi alla tecnica del complemento per eseguire una somma algebrica?

Perché per il computer è piú semplice calcolare il complemento ad una certa base delle cifre di un numero che non ricorrere al noto criterio aritmetico dei numeri relativi, criterio che, come si sa, consiste nel confrontare i valori assoluti dei due valori numerici prima che questi possano essere operati algebricamente.

Infatti, la somma algebrica di numeri relativi (dotati di segno) è un numero relativo con lo stesso segno del numero relativo che ha maggior valore assoluto e con valore pari al valore assoluto della somma algebrica dei due numeri relativi.

Per esempio, la somma algebrica tra i due numeri:

+40 e −80

è un numero relativo con segno − (meno), cioè con lo stesso segno di −80, che è maggiore in valore assoluto a +40, e con valore pari al valore assoluto della somma algebrica (o differenza relativa):

80−40=40

ossia si ottiene il numero relativo −40.

Analogamente:

+3−47

è un numero relativo con segno − (meno), cioè con lo stesso segno di −47, che risulta in valore assoluto maggiore di +3, e con valore:

47−3=44

cioè il numero relativo −44.

Il risultato della somma algebrica:

$$+40-31$$

è invece un numero relativo con segno + (piú), cioè con lo stesso segno di +40 che risulta in valore assoluto maggiore di −31, e con valore dato da:

$$40-31=9$$

cioè il numero relativo +9.

Similmente il risultato della somma algebrica:

$$-43-54$$

è un numero relativo con segno − (meno), cioè con il segno di entrambi i numeri relativi, e con valore dato da:

$$43+54=97$$

cioè il numero relativo −97.

Come si vede, le note regole aritmetiche richiedono operazioni logiche (confronti) ed operazioni piú propriamente aritmetiche (somme e sottrazioni), mentre la tecnica del complemento (utilizzata dal computer) richiede solamente operazioni di addizione e complementazione.

Quest'ultima, tra l'altro, risulta veramente semplice da eseguire: per calcolare il complemento ad una certa base di un numero non occorre fare una sottrazione, ma basta calcolare i complementi di ciascuna cifra e sottrarre una unità (1), ad eccezione che nel caso della cifra meno significativa.

Per esempio, il complemento alla base 100 del numero:

$$87$$

si ottiene complementando alla base (10) la cifra 8 da cui si ottiene 2 che, diminuito di 1, diventa 1, e complementando sempre alla stessa

base (10) la cifra 7, ottenendo 3 che non va diminuito di 1 perché si tratta della cifra meno significativa.

Il complemento a 100 di 87 è quindi 13 (valore che risulta dalla differenza fra 100 e 87).

Analogamente il complemento a 1000 di 753 si ottiene complementando a 10 le singole cifre e decurtando una unità da ogni risultato, ottenendo cioè il numero:

247

(che corrisponde alla differenza tra i numeri 1000 e 753).

Nel caso si debba calcolare la somma algebrica di due numeri negativi, per esempio −13−9, si procede in modo analogo, tenendo presente che l'operazione di complementazione va saldata, come si dice, cioè occorre eseguire alla fine sul risultato tanti complementi quanti ne sono stati effettuati durante l'operazione di somma fra i singoli addendi.

Quindi, per calcolare con la tecnica del complemento la somma algebrica:

−13−9

si procede nel modo descritto di seguito.

1) Si complementano entrambi gli addendi (negativi) −13 e −9, ottenendo i numeri:

87 (complemento a 100 di −13)
1 (complemento a 10 di −9)

poi si sommano 87 e 1, per cui si ottiene:

87+
1=
88

2) Siccome sono stati effettuati due complementi al passo 1, si deve ricomplementare due volte (una prima volta con la base −100 e una seconda volta con la base −10) il risultato ottenuto:

COMP (88) = −12 (COMP = COMPLEMENTO)
(−100)

COMP (−12) = −22
(−10)

(quest'ultimo è il risultato della somma algebrica −13−9).

Se si indicano con:

$A = -13$
$B = -9$
$B_1 = 100$
$B_2 = 10$

le operazioni 1) e 2) precedentemente considerate equivalgono alle due seguenti espressioni:

1*) $B_1 + A + B_2 + B = 100 - 13 + 10 - 9 = 88$

2*) $-B_1 + B_1 + A + B_2 + B - B_2 = -100 + 100 - 13 + 10 - 9 - 10 = -22$

L'espressione 2*) dimostra infatti che le basi vengono aggiunte e sottratte artificialmente per trasformare le sottrazioni in somme (di complementi).

Si vedrà ora come si procede per effettuare operazioni algebriche con i numeri binari.

Si voglia, per esempio, eseguire la sottrazione tra i numeri binari 10 e 11 (che, come precedentemente visto, sono i codici binari dei numeri decimali 2 e 3).

Si tratta di sottrarre 11 da 10; per fare ciò si adotta il seguente procedimento.

1) Innanzitutto si complementa il numero binario negativo −11 alla base 100 (equivalente al numero decimale 4); in pratica si tratta di eseguire la differenza (con la tecnica del complemento) tra i numeri 100 e 11, cioè di complementare i singoli bit del sottraendo 11.

   Il risultato di questo complemento è 01 (equivalente al numero decimale 1), per ottenere il quale si complementa ogni cifra bi-

naria del numero 11 alla base 10 e poi si sottrae 1 (ad eccezione del bit meno significativo). Quindi, il complemento di 1 a 10 è 1 che, diminuito di 1, diventa 0 (prima cifra binaria di 01), mentre il complemento della cifra meno significativa di 11 (cioè 1) al numero 10 è 1 (seconda cifra binaria di 01).

Dopo aver ottenuto il numero binario 01 (che, come si è visto, è il complemento del numero 11 a 100) occorre addizionarlo al primo addendo 10 (2 decimale) per ottenere:

$$\begin{array}{r} 01+ \\ \underline{10=} \\ 11 \end{array}$$

cioè il numero binario 11 (equivalente al decimale 3).

2) Con lo stesso criterio usato per il sistema decimale, si deve ora ricomplementare il risultato 11 alla base 100, cioè calcolare la differenza +11 −100 con la tecnica del complemento.

Anche in questo caso si complementano le singole cifre binarie di 11 alla base 10.

Si ottiene cosí il numero negativo binario:

−01

equivalente al numero −1 del sistema decimale.

In questo sistema la differenza:

2−3

risulta infatti −1.

d) Moltiplicazione e divisione

La moltiplicazione tra numeri del sistema decimale equivale, come è noto, ad una serie di somme; queste saranno tante quante ne indica il valore del numero moltiplicatore.

Per esempio, il prodotto seguente:

2×3

equivale a sommare tante volte il 2 quante ne indica il valore del moltiplicatore 3, cioè:

$$2\times3=2+2+2=6$$

Analogamente,

$$6\times4=6+6+6+6=24$$

Nel caso in cui il moltiplicatore superi le due cifre decimali, occorre predisporre gli operandi in colonna e seguire le note regole dell'aritmetica per calcolare la moltiplicazione.

Per esempio, per calcolare il prodotto:

$$12\times24$$

sarebbe oltremodo prolisso sommare 24 volte il numero 12. Si ricorre quindi al noto criterio che consiste nell'eseguire tanti prodotti parziali tra il moltiplicando 12 e le cifre decimali (4 e 2) del moltiplicatore e nel sommare alla fine tra loro i prodotti parziali incolonnati in forma scalare:

$$\begin{array}{r} 12\times \\ 24= \\ \hline 48 \\ 24\text{ -} \\ \hline 288 \end{array}$$

Se inoltre gli operandi sono dotati di cifre decimali, alla fine dell'operazione di moltiplicazione si staccheranno sul risultato tante cifre decimali quante sono complessivamente le cifre decimali del moltiplicando e del moltiplicatore.

Per esempio:

$$\begin{array}{r} 2{,}1\times \\ 3{,}3= \\ \hline 63 \\ 63\text{ -} \\ \hline 6{,}93 \end{array}$$

Quando il moltiplicatore contiene degli zeri, l'operazione di molti-

plicazione si può ulteriormente semplificare spostandosi di tante colonne a sinistra quanti sono gli zeri del moltiplicatore.

Ad esempio, 12×4009 si può calcolare nel seguente modo:

```
   12×
 4009=
-----
  108
 48---
-----
 48108
```

La moltiplicazione tra due numeri binari si esegue in maniera analoga a quanto visto per i numeri espressi nel sistema decimale, tenendo presenti i seguenti casi elementari:

$$0\times0=0$$
$$0\times1=0$$
$$1\times0=0$$
$$1\times1=1$$

Occorre sottolineare che il calcolo del segno nelle operazioni tra numeri relativi segue regole ben precise, che nel caso dell'elaborazione con il computer sono concretizzate con il posizionamento del bit 1 (per numeri negativi) e del bit 0 (per numeri positivi) nella parte piú a sinistra del valore numerico.

Si voglia, per esempio, moltiplicare il numero binario 1101 per il numero binario 101; l'operazione risulta la seguente:

```
  1101×
   101=
-------
  1101
 1101--
-------
1000001
```

Analogamente, il prodotto 111×1001 si calcolerà cosí:

```
   111×
  1001=
-------
   111
 111---
-------
111111
```

Si vedrà ora come si effettua la divisione tra due numeri.

Nel sistema decimale tale operazione si esegue dividendo gradualmente le cifre o gruppi di cifre del dividendo per il valore del divisore.

Le cifre del dividendo sono ricavate aggiungendo di volta in volta ai resti parziali le successive cifre del numero da dividere.

Ad esempio, per dividere 1240 per 12 si effettua prima la divisione di 12 (prime due cifre di 1240) per 12 (divisore). Il risultato di questa prima divisione è 1 (che costituisce la prima cifra piú significativa del quoziente).

Quindi si moltiplica 1 per il divisore 12 e si sottrae il risultato (12) dalle prime due cifre del dividendo, ottenendo come resto di quest'ultima operazione 0 (zero).

A questo valore del resto si aggiunge un'altra cifra del dividendo (4) e si prosegue la divisione come prima, considerando come dividendo nuovo quello appena ottenuto (cioè 4).

L'operazione di divisione finisce quando non ci sono piú cifre del dividendo da aggiungere ai resti parziali.

Operativamente la divisione tra 1240 e 12 si esegue nel seguente modo:

```
1240 | 12
12   | 103
---
 04
  0
 ---
 40
 36
 ---
 =4
```

Si ottiene cosí il quoziente 103 e il resto 4.

Se poi si vuole continuare per valori decimali, occorre mettere una virgola dopo la cifra meno significativa dell'ultimo quoziente calcolato ed aggiungere uno 0 (zero) al resto, considerando quest'ultimo il nuovo dividendo.

Questo metodo, normalmente utilizzato per eseguire una divisione tra due numeri, non è praticabile dal calcolatore, il quale ne usa un altro, analogamente a quanto si è visto per la moltiplicazione.

Infatti, come per moltiplicare due numeri si addiziona tante volte il primo operando quante ne indica il valore del secondo, cosí per dividere un numero per un altro si sottrae tante volte quest'ultimo dal primo fino ad avere come resto delle successive sottrazioni un valore minore del divisore.

In base a questo criterio, il quoziente della divisione sarà dato dal numero di sottrazioni che si sono effettuate, ed il resto sarà quello dell'ultima sottrazione.

Per esempio, nel sistema decimale, per dividere il numero 100 per 12 si eseguono successivamente le seguenti sottrazioni:

$$\begin{aligned}
100-12&=88\\
88-12&=76\\
76-12&=64\\
64-12&=52\\
52-12&=40\\
40-12&=28\\
28-12&=16\\
16-12&=\boxed{4} \qquad \text{minore di } 12
\end{aligned}$$

Il risultato o quoziente della divisione tra 100 e 12 è quindi 8 (numero delle sottrazioni che sono state eseguite) con resto 4.

La divisione tra numeri binari si effettua proprio in questo modo, cioè tramite una successione di sottrazioni con la tecnica del complemento.

Per esempio, per dividere il numero binario 1101 per il numero binario 11, si eseguono successivamente le seguenti sottrazioni:

$$\begin{aligned}
1101-11&=1101+(01)^c=(1110)^c=1010\\
1010-11&=1010+(01)^c=(1011)^c=\ 111\\
\ 111-11&=\ 111+(01)^c=(1000)^c=\ 100\\
\ 100-11&=\ 100+(01)^c=(101)^c=\ \boxed{01} \qquad \text{minore di } 11
\end{aligned}$$

Il risultato o quoziente della divisione tra il numero binario 1101 e il numero binario 11 è quindi 100 (cioè 4 nel sistema decimale, equiva-

lente al numero di sottrazioni effettuate), con resto 01.

Si noti come nelle precedenti operazioni si è indicata con la lettera "c" minuscola un'operazione di complementazione (al valore binario 100, in questo caso, che una volta va sommato ed una volta va sottratto per ottenere il risultato finale di ogni sottrazione).

Come ulteriore esempio, si esegue ora la divisione tra i numeri binari 111 (7 decimale) e 10 (2 decimale):

$$111-10=111+(10)^c=(1001)^c=101$$
$$101-10=101+(10)^c=(111)^c=\ 11$$
$$11-10=\ 11+(10)^c=(101)^c=\boxed{01}\quad \text{minore di } 10$$

Il risultato di questa divisione è 11 (pari al numero 3 di sottrazioni effettuate), con resto 01 (corrispondente a 1 decimale).

3. – Variabili e funzioni

La terminologia usata in informatica per le variabili e le funzioni è strettamente collegata a quella adottata in matematica.

Il computer parla la stessa lingua della macchina

Le definizioni di variabili e funzioni sono quindi analoghe alle definizioni matematiche. Si deve però tenere presente che in informatica non soltanto i valori numerici sono oggetto di elaborazione, ma anche i caratteri alfanumerici.

La **variabile** quindi, nell'ambito della scienza dei calcolatori, è da intendersi una **"entità" numerica o alfanumerica** che, nel corso di un'elaborazione, può assumere un qualsiasi valore indipendente dal valore di altre variabili.

La **funzione**, invece, è un'**entità sempre o numerica o alfanumerica**, il cui valore dipende (è funzione di...) da una o piú variabili indipendenti; la funzione si dice anche **variabile dipendente**.

Per esempio, la quantità di nome X, che assume nel corso di un programma i tre valori:

1, 5, 3.14

indipendenti tra loro e da qualsiasi altra variabile, è definita "**variabile indipendente**", mentre la quantità di nome Y, che assume nel corso dello stesso programma i tre valori:

2, 10, 6.28

che risultano doppi rispetto ai tre precedenti valori di X, è definita "funzione", perché i suoi valori "dipendono" dai valori di un'altra variabile (la X appunto).

Diremo quindi che **due grandezze variabili sono dipendenti quando al variare dell'una varia anche l'altra.**

In matematica, per indicare che Y è funzione di X, si scrive:

1) $Y = f(X)$ (si legge "Y uguale effe di X")

dove Y è la denominazione usata per la variabile dipendente e $f(X)$ è il simbolo di funzione dipendente.

Se si scrive ad esempio $Y = 2X$, il termine 2X è la forma esplicita di tale funzione (un'espressione algebrica piú o meno complessa, in cui appare la variabile indipendente).

Se una variabile dipendente Y dovesse dipendere da piú di una variabile indipendente (ad esempio, dalle due variabili X e Z), si dovrebbe scrivere:

2) $Y = f(X,Z) = 2X + 3Z$

In questo caso Y dipenderebbe dalla somma del doppio di X e del triplo di Z.

Il simbolo $f(X, Z)$ riporta sempre tra parentesi i nomi delle variabili indipendenti (dette anche **argomenti della funzione**).

La funzione 1) $Y = 2X$ rappresenta geometricamente l'equazione di una retta che passa per l'origine degli assi cartesiani, mentre la funzione 2) $Y = 2X + 3Z$ rappresenta l'equazione di un piano: tramite questa funzione cioè vengono dati tutti i punti di coordinate Y in funzione delle altre due coordinate indipendenti X e Z.

Le raffigurazioni geometriche delle due funzioni 1) e 2) sono riportate rispettivamente nella *fig. 3* e nella *fig. 4*.

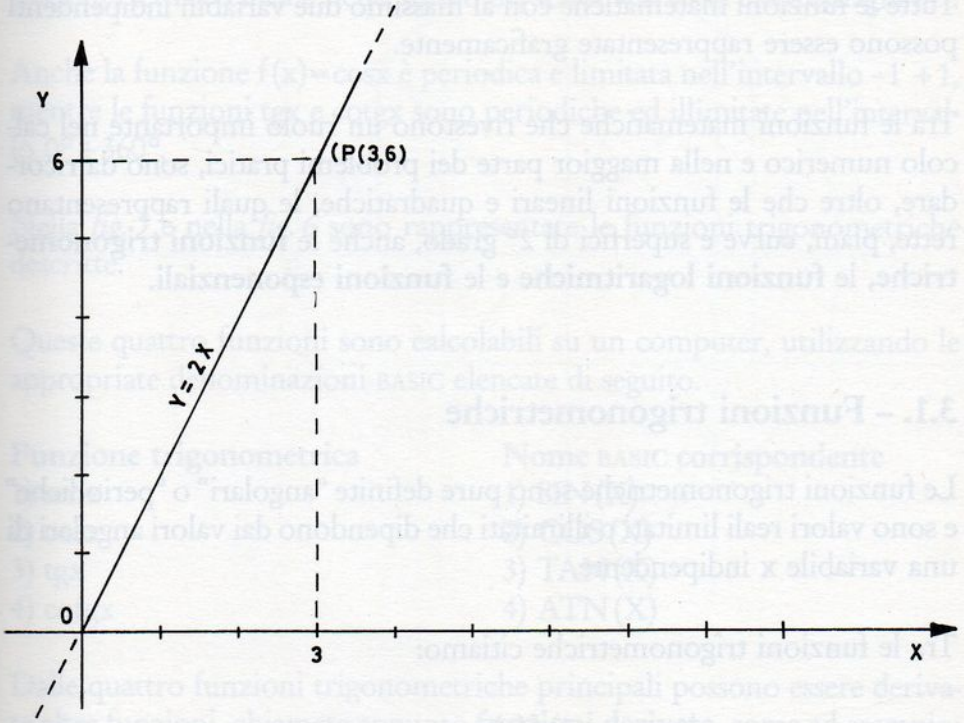


Fig. 3 – Rappresentazione grafica della funzione Y=2X, retta di un piano passante per l'origine e per il punto P(3,6).

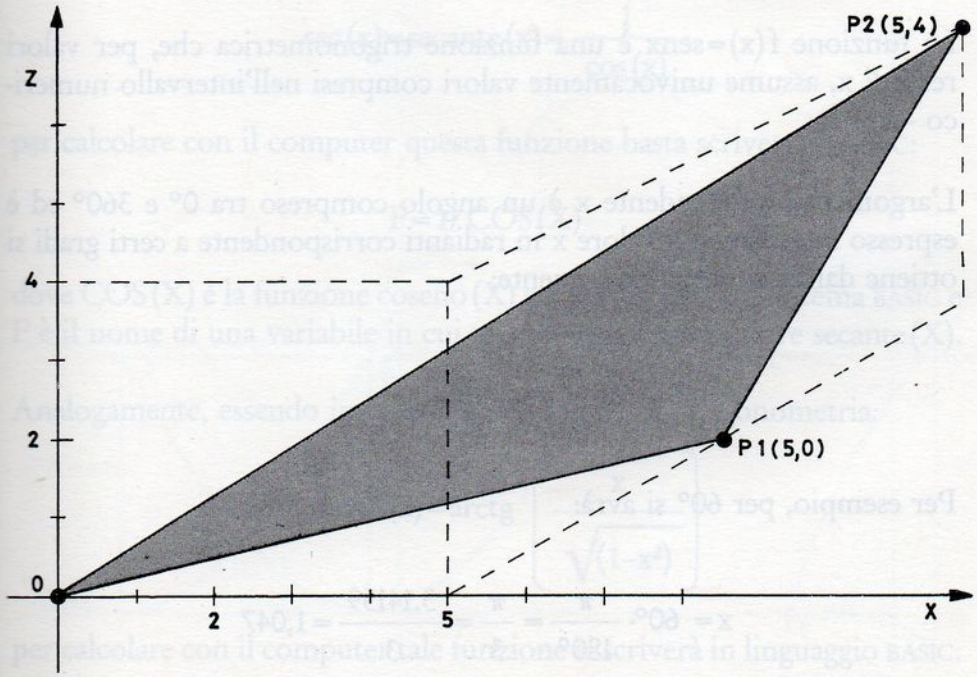


Fig. 4 – Rappresentazione grafica della funzione Y=2X+3Z, superficie del piano passante per l'origine degli assi O(0,0) e per i punti P1(5,0) e P2(5,4).

Tutte le funzioni matematiche con al massimo due variabili indipendenti possono essere rappresentate graficamente.

Tra le funzioni matematiche che rivestono un ruolo importante nel calcolo numerico e nella maggior parte dei problemi pratici, sono da ricordare, oltre che le funzioni lineari e quadratiche, le quali rappresentano rette, piani, curve e superfici di 2° grado, anche le **funzioni trigonometriche,** le **funzioni logaritmiche** e le **funzioni esponenziali.**

3.1. - Funzioni trigonometriche

Le funzioni angolari periodiche

Le funzioni trigonometriche sono pure definite "angolari" o "periodiche" e sono valori reali limitati o illimitati che dipendono dai valori angolari di una variabile x indipendente.

Tra le funzioni trigonometriche citiamo:

1) $f(x)=\text{sen}x$
2) $f(x)=\cos x$
3) $f(x)=\text{tg}x$
4) $f(x)=\text{cotg}x$

La funzione $f(x)=\text{sen}x$ è una funzione trigonometrica che, per valori reali di x, assume univocamente valori compresi nell'intervallo numerico -1 $+1$.

L'argomento indipendente x è un angolo compreso tra 0° e 360° ed è espresso in radianti. Il valore x in radianti corrispondente a certi gradi si ottiene dalla proporzione seguente:

$$\frac{x}{\text{gradi}} = \frac{2\pi}{360°} = \frac{\pi}{180°}.$$

Per esempio, per 60° si avrà:

$$x = 60° \cdot \frac{\pi}{180°} = \frac{\pi}{3} = \frac{3.14159}{3} = 1{,}047$$

La funzione $f(x)=\text{sen}x$ è periodica e limitata: dopo un angolo di 360° cioè si ripete con uguali valori compresi tra i limiti -1 e $+1$. Essa è molto

usata in elettrotecnica e nello studio dei fenomeni fisici elettromagnetici.

Anche la funzione f(x)=cosx è periodica e limitata nell'intervallo −1 +1, mentre le funzioni tgx e cotgx sono periodiche ed illimitate nell'intervallo 0°÷360°.

Nella *fig. 5* e nella *fig. 6* sono rappresentate le funzioni trigonometriche descritte.

Queste quattro funzioni sono calcolabili su un computer, utilizzando le appropriate denominazioni BASIC elencate di seguito.

Funzione trigonometrica	**Nome BASIC corrispondente**
1) senx	1) SIN(X)
2) cosx	2) COS(X)
3) tgx	3) TAN(X)
4) cotgx	4) ATN(X)

Dalle quattro funzioni trigonometriche principali possono essere derivate altre funzioni, chiamate appunto **funzioni derivate**, come ad esempio: sec(x), cosec(x), arsen(x), arcos(x), arctg(x) ecc.

Per esempio, essendo in base alle proprietà della trigonometria

$$\sec(x)=\text{secante}(x)=\frac{1}{\cos(x)},$$

per calcolare con il computer questa funzione basta scrivere in BASIC:

```
F=1/COS(X)
```

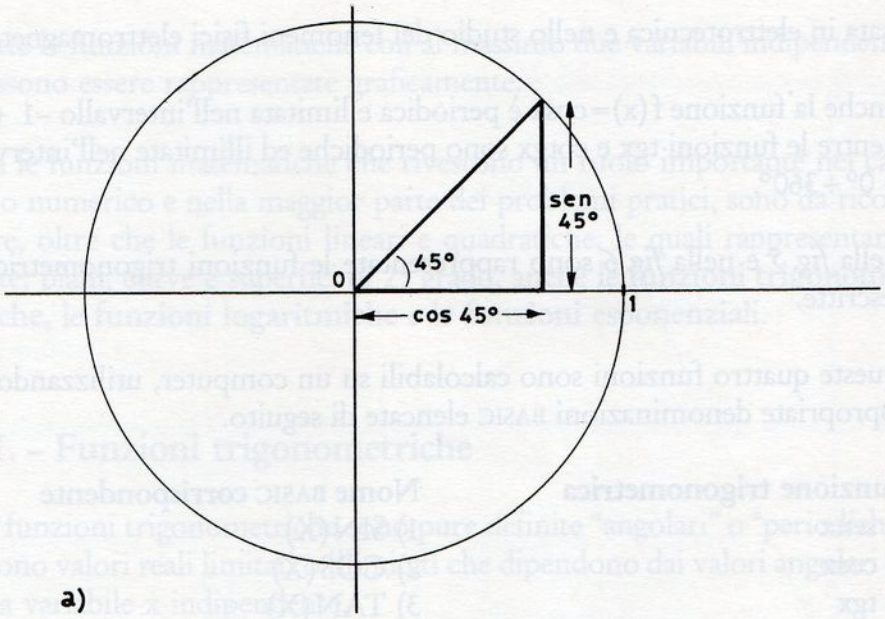
dove COS(X) è la funzione coseno(X) già predisposta nel sistema BASIC e F è il nome di una variabile in cui va memorizzato il valore secante(X).

Analogamente, essendo in base alle proprietà della trigonometria:

$$\text{arsen}(x)=\text{arctg}\left[\frac{x}{\sqrt{(1-x^2)}}\right]$$

per calcolare con il computer tale funzione si scriverà in linguaggio BASIC:

```
F=ATN(X/SQR(-X*X+1)).
```



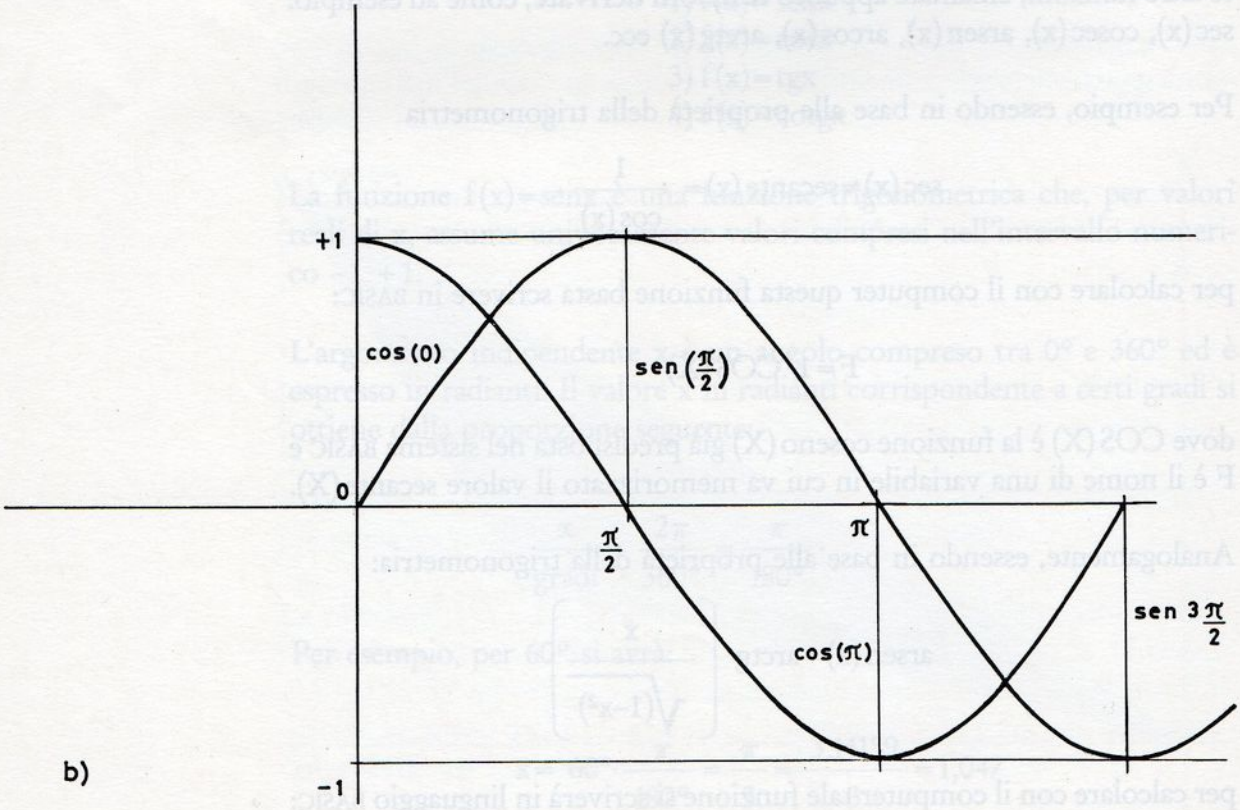
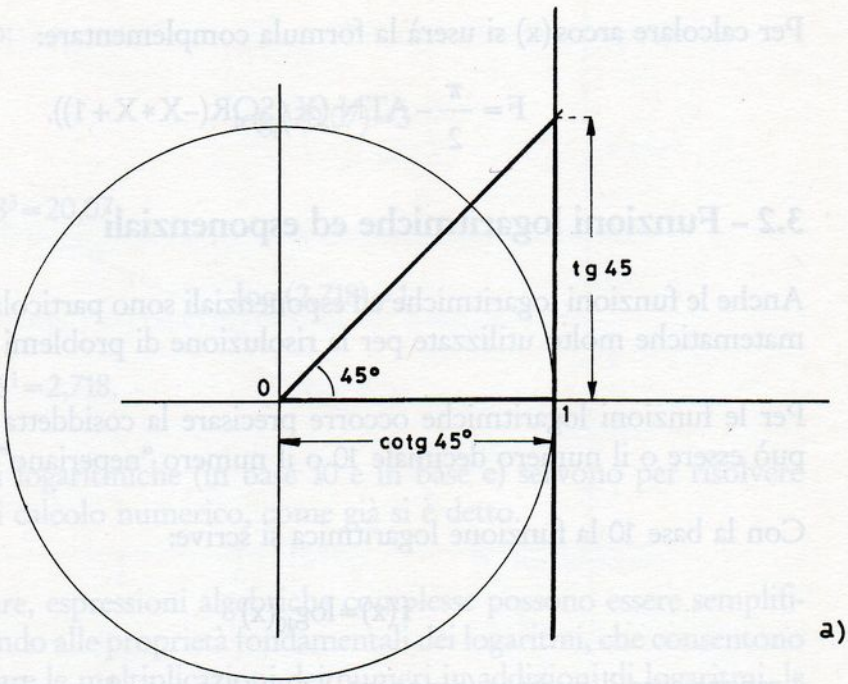
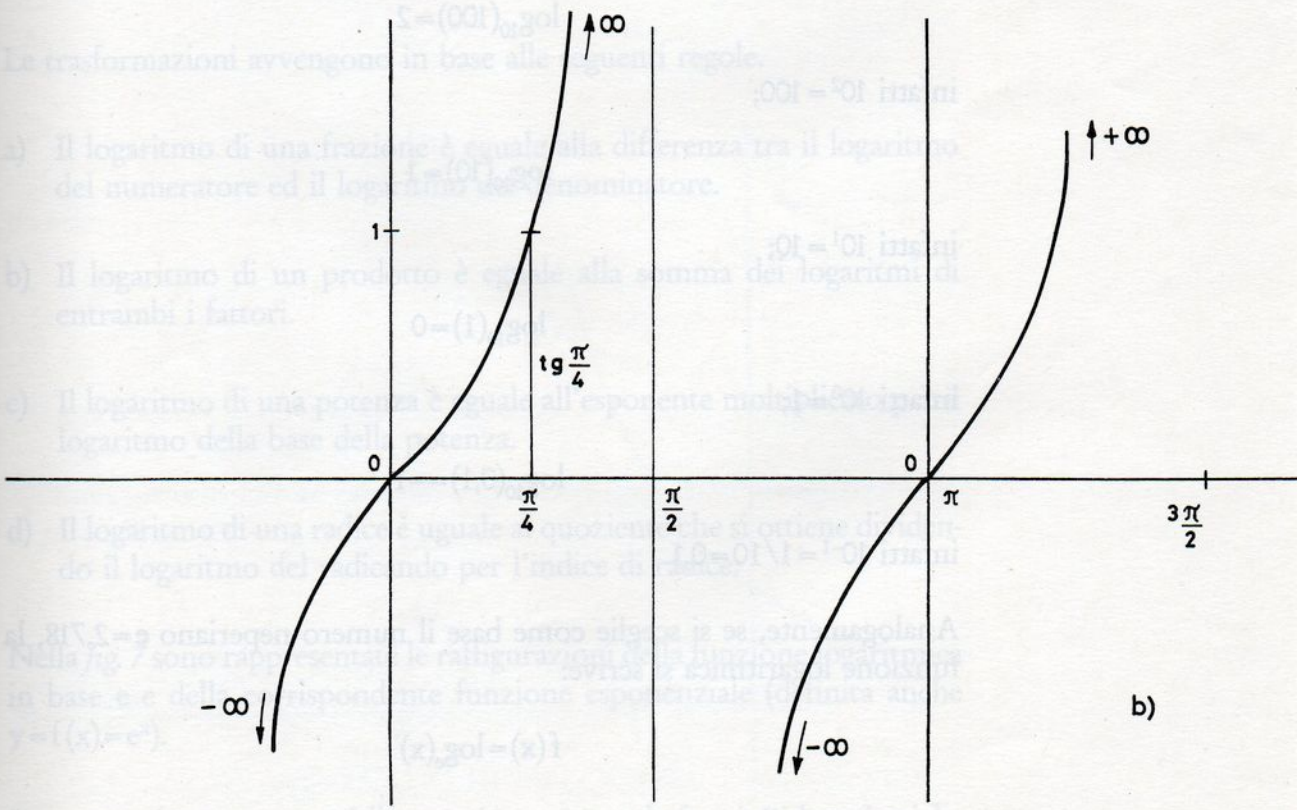


Fig. 5 – Rappresentazione delle funzioni sen (x) e cos (x) sulla circonferenza trigonometrica (a) e loro rappresentazione su un sistema di assi cartesiani ortogonali (b).



a)



b)

Fig. 6 – Rappresentazione delle funzioni tg(x) e cotg(x) sulla circonferenza trigonometrica (a) e loro rappresentazione su un sistema di assi cartesiani (b).

Per calcolare arcos(x) si userà la formula complementare:

$$F = \frac{\pi}{2} - ATN\ (X/SQR(-X*X+1)).$$

3.2 - Funzioni logaritmiche ed esponenziali

Anche le funzioni logaritmiche ed esponenziali sono particolari funzioni matematiche molto utilizzate per la risoluzione di problemi numerici.

Per le funzioni logaritmiche occorre precisare la cosiddetta "base", che può essere o il numero decimale 10 o il numero "neperiano" 2,718.

Con la base 10 la funzione logaritmica si scrive:

$$f(x) = \log_{10}(x)$$

e rappresenta il numero (esponente) a cui occorre elevare la base 10 per avere il valore x. Per esempio:

$$\log_{10}(100) = 2$$

infatti $10^2 = 100$;

$$\log_{10}(10) = 1$$

infatti $10^1 = 10$;

$$\log_{10}(1) = 0$$

infatti $10^0 = 1$;

$$\log_{10}(0{,}1) = -1$$

infatti $10^{-1} = 1/10 = 0{,}1$.

Analogamente, se si sceglie come base il numero neperiano **e**=2,718, la funzione logaritmica si scrive:

$$f(x) = \log_e(x)$$

e rappresenta il numero a cui occorre elevare la base **e**=2,718 per ottenere il valore x.

Per esempio:

$$\log_e(20{,}07) = 3$$

perché $2{,}718^3 = 20{,}07$;

$$\log_e(2{,}718) = 1$$

perché $2{,}718^1 = 2{,}718$.

Le funzioni logaritmiche (in base 10 e in base **e**) servono per risolvere problemi di calcolo numerico, come già si è detto.

In particolare, espressioni algebriche complesse possono essere semplificate ricorrendo alle proprietà fondamentali dei logaritmi, che consentono di trasformare le moltiplicazioni dei numeri in addizioni di logaritmi, le divisioni dei numeri in sottrazioni di numeri, le elevazioni a potenza in moltiplicazioni e le estrazioni di radici in divisioni.

Le trasformazioni avvengono in base alle seguenti regole.

a) Il logaritmo di una frazione è eguale alla differenza tra il logaritmo del numeratore ed il logaritmo del denominatore.

b) Il logaritmo di un prodotto è eguale alla somma dei logaritmi di entrambi i fattori.

c) Il logaritmo di una potenza è eguale all'esponente moltiplicato per il logaritmo della base della potenza.

d) Il logaritmo di una radice è uguale al quoziente che si ottiene dividendo il logaritmo del radicando per l'indice di radice.

Nella *fig. 7* sono rappresentate le raffigurazioni della funzione logaritmica in base **e** e della corrispondente funzione esponenziale (definita anche $y = f(x) = e^x$).

La nomenclatura usata nel linguaggio BASIC per le funzioni logaritmiche ed esponenziali è la seguente.

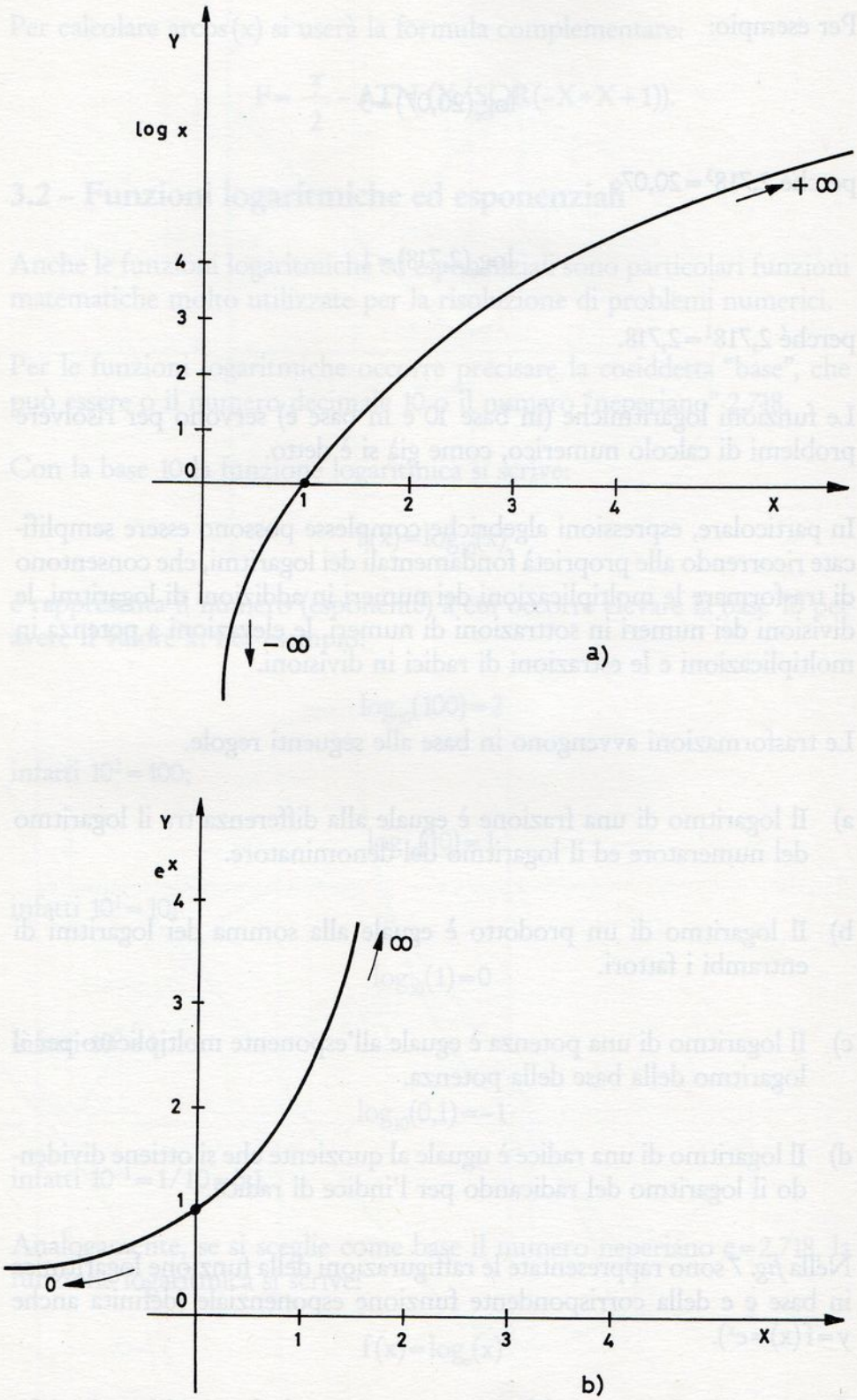


Fig. 7 – Rappresentazione geometrica della funzione logaritmica in base e (a) e della funzione esponenziale (b).

Espressione matematica	Nome BASIC corrispondente
$\log_e(x)$	LOG(X)
e^x	EXP(X)

Per calcolare il $\log_{10}x$ si ricorre alla trasformazione seguente.

Posta l'incognita y da determinare uguale a $\log_{10}x$, cioè:

$$y=\log_{10}(x)$$

si ha:

$$10^Y=x \qquad \text{(per definizione)}$$

Si prendano ora i logaritmi in base **e** di entrambi i membri della precedente relazione:

$$\log_e 10^Y=\log_e x$$

cioè:

$$y \log_e 10=\log_e x$$

da cui:

$$y=\frac{\log_e x}{\log_e 10}$$

che può essere calcolata con il computer.

Quindi, per calcolare ad esempio con il linguaggio BASIC il seguente logaritmo in base 10:

$$\log_{10}3.14159$$

si scriverà:

```
Y=LOG (3.14159)/LOG(10)
```

da cui si ottiene: .497149506, che è appunto il logaritmo di 3.14159.

Per calcolare il logaritmo in base 10 del numero 58, si scriverà:

```
Y=LOG(58)/LOG(10)
```

da cui si ottiene: 1,74819.

Anche dalle funzioni logaritmiche ed esponenziali possono essere derivate molte altre funzioni. Tra le funzioni derivate piú note e piú usate ricordiamo il seno e il coseno iperbolico (senh e cosh).

Per la funzione senh(x) si ha:

$$\text{senh}(x) = \frac{e^{x} - e^{-x}}{2}$$

e per la funzione cosh(x):

$$\cosh(x) = \frac{e^{x} + e^{-x}}{2}$$

Per calcolare con il computer la funzione senh(x) si scriverà:

```
F=(EXP(X)-EXP(-X))/2.
```

Per calcolare la funzione cosh(x) si scriverà:

```
F=(EXP(X)+EXP(-X))/2.
```

Infine , per calcolare le funzioni inverse arsenh(x) e arcosh(x) si scriverà rispettivamente:

```
F=LOG(X+SQR(X*X+1))
```

```
F=LOG(X+SQR(X*X-1))
```

Nella lezione *Analisi e programmazione 11* saranno trattate speciali funzioni, dette anche **funzioni di computer**, le quali consentono la manipolazione e l'elaborazione delle stringhe alfanumeriche.

Con tali funzioni è possibile operare sui singoli caratteri che compongono una parola o gruppi di parole, eseguire permutazioni, combinazioni di gruppi di lettere, confrontare ed ordinare elenchi di nomi, ecc.

Mediante l'utilizzo delle funzioni di computer la parola diventa così un'entità di elaborazione operabile in modo analogo ai numeri.

4. - Probabilità e statistica con il computer

Come si è visto nel precedente capitolo, il calcolo numerico di espressioni matematiche contenenti funzioni matematiche viene risolto nel computer utilizzando e derivando adeguatamente alcune funzioni base, quali le funzioni trigonometriche, logaritmiche ed esponenziali, che sono già programmate all'interno del linguaggio BASIC.

Per chi ha un po' di conoscenza di analisi numerica, le funzioni base sono programmate a guisa di "sviluppi in serie", avendo ogni serie determinati coefficienti archiviati all'interno delle memorie RAM o ROM del computer.

Quando al computer si chiede, attraverso il programma, di calcolare per esempio e^3, quest'ultimo provvede a richiamare speciali coefficienti, che sono il risultato dello sviluppo in serie di potenze della funzione matematica:

$$f(x)=e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^{n-1}}{n!}+\frac{x^n}{n!}e^{\theta x}$$

dove x è l'argomento indipendente fornito dal programma (nel nostro caso, x=3).

Questi coefficienti (che per la funzione e^x sono 1, 1, 1/2, 1/6, 1/24,...) vengono poi moltiplicati e combinati con le potenze successive dell'argomento x, in modo che venga calcolata la somma algebrica, per x=3:

$$f(x)=e^x=e^3=1+3+\frac{3^2}{2}+\frac{3^3}{6}+\frac{3^4}{24}+\ldots..R_u$$

R_u è la rimanenza che si ottiene calcolando la serie fino al termine $x^{n-1}/n!$ Il calcolo dello sviluppo sopra riportato fino al quattordicesimo termine fornisce il valore e^3 preciso fino alla decima cifra decimale.

Il punto esclamativo (!) posto dopo un numero significa che esso è un fattoriale, ossia rappresenta quel numero moltiplicato per ogni numero minore di esso. Ad esempio, 3!=3·2·1.

Quasi tutte le funzioni matematiche trigonometriche logaritmiche ed esponenziali sono calcolate con sviluppi in serie.

Per quanto riguarda il calcolo delle probabilità e il calcolo statistico, il computer mette a disposizione una sola funzione (come abbiamo già visto nelle precedenti lezioni), la quale consente di ottenere numeri casuali: la funzione RND (X).

Per un determinato valore di X (che può essere nullo, positivo oppure negativo), la funzione RND (X) estrae un numero a caso compreso tra Ø e 1.

In pratica, utilizzando questa funzione è come se si estraesse "a caso" da un'urna un qualsiasi numero, il cui valore è nell'intervallo

$$Ø \div 1$$

Numeri casuali, funzioni trigonometriche, logaritmiche ed esponenziali costituiscono il supporto base matematico da utilizzare per effettuare elaborazioni numeriche e statistiche.

Nei grossi sistemi di elaborazione, a fianco dei potenti linguaggi di elaborazione tipo FORTRAN, BASIC, COBOL, esistono complesse "librerie" di programmi matematici e statistici, in grado di calcolare complicate funzioni derivate.

Per richiamare ed utilizzare questi programmi basta solamente scrivere in uno dei linguaggi simbolici citati prima un'istruzione di "salto" ad una subroutine archiviata nella libreria matematica, seguita dall'elenco di nomi di costanti o variabili che verranno poi utilizzati dal programma di libreria.

I piccoli sistemi di elaborazione (home e personal computer) non sono dotati di tali programmi, a causa delle loro limitate capacità di memoria; da un po' di tempo però stanno per essere immessi sul mercato microsistemi dotati di maggiori possibilità elaborative.

In ogni caso, cioè sia con piccoli sia con grandi sistemi di elaborazione, per effettuare calcoli matematici e statistici occorre avere una certa conoscenza e una certa pratica di "analisi numerica" e di "probabilità e statistica"; il calcolatore serve soltanto come mezzo di calcolo per le elaborazioni dei problemi.

5. - L'algebra di Boole

Il calcolatore digitale, come si è visto in una delle prime lezioni, può elaborare soltanto due cifre binarie: 0 e 1.

I programmi ed i dati introdotti all'inizio dell'elaborazione sono trasformati, da altri programmi, in gruppi organizzati e delimitati di cifre binarie.

Che cosa rende possibile al computer riconoscere, prelevare, trattare i caratteri 0 e 1?

Che cosa gli consente di distinguere caratteri che rappresentano istruzioni da caratteri che rappresentano dati dell'elaborazione?

La risposta si può trovare osservando la parte interna circuitale del computer stesso e cercando per un momento di immaginare quest'ultima come una metropoli in miniatura vista di notte, nella quale una miriade di segnali luminosi pulsano ad un ritmo frenetico.

I circuiti elettronici sono come le strade di una città su cui scorrono treni e treni di impulsi elettrici (non luci ma stati di tensione elettrica, distinti dai valori convenzionali 0 e 1); essi si combinano tra loro e generano altri impulsi binari.

Ora l'insieme delle regole e delle operazioni logiche con cui i circuiti elettronici operano tra loro con combinazioni, spostamenti, raffronti di segnali elettrici-binari segue una logica di calcolo di tipo matematico, che va sotto il nome di **algebra delle proposizioni** o **algebra booleana**, dal nome del logico irlandese George Boole che per primo ne formulò nel XIX secolo le basi teoriche.

L'algebra booleana si era già rivelata utile nello studio degli orari ferroviari e delle reti telefoniche, poiché consentiva di organizzare sistematicamente i traffici delle rispettive linee.

Con l'algebra booleana apparve subito piú facile determinare le configurazioni circuitali del calcolatore, vedendo quest'ultimo come una rete di circuiti binari, tutti uguali fra loro.

L'algebra booleana è quindi un insieme finito di elementi **a**, **b**, **c**, ..., che

possono avere due soli valori: 0,1, ed è anche il complesso delle operazioni logiche con cui tali valori sono combinati tra loro.

Le operazioni logiche dell'algebra booleana sono tre:

- addizione (operazione OR);
- prodotto (operazione AND);
- inversione (operazione NOT).

Queste operazioni logiche sono definite come di seguito precisato.

ADDIZIONE (OPERAZIONE OR)

Operazione OR

La somma di due elementi **a, b** dell'insieme (a due soli valori, 0 e 1) è un elemento **c** dell'insieme (cioè ancora un elemento di valore 0 oppure 1) che assume valore 1 se **almeno** uno dei valori di **a** o **b** è 1.

Nella tabella della *fig. 8*, detta **tabella della verità**, è riportato lo schema logico-operativo dell'addizione.

	ELEMENTI OPERANDI		ELEMENTO RISULTATO
	a	**b**	**c**
1)	0	1	1
2)	1	0	1
3)	1	1	1
4)	0	0	0

Fig. 8 - Tabella della verità dell'operazione OR (addizione).

Osservando tale tabella si rileva che:

— nel caso 1), **c** vale 1 perché "almeno" **b** vale 1;
— nel caso 2), **c** vale 1 perché "almeno" **a** vale 1;
— nel caso 3), **c** vale 1 perché **a, b** valgono entrambi 1;
— nel caso 4), **c** vale 0 perché né **a** né **b** valgono 1.

Il nome OR dato a questa operazione deriva dalla lingua inglese, nella quale la congiunzione "o" viene appunto denominata OR: infatti, **c** assume valore 1 se **almeno** uno, ma non necessariamente uno solo, dei valori di **a** o **b** è 1.

Il segno grafico del circuito elettronico che svolge la funzione OR a due

ingressi (**a** e **b**) con il simbolo della rispettiva funzione logica è rappresentato nella *fig. 9*.

Fig. 9 – Segno grafico del circuito elettronico che svolge la funzione OR.

Si può anche dire che il circuito OR attribuisce il valore 1 a **c** se **a** o **b** vale 1.

PRODOTTO (OPERAZIONE AND)

Operazione AND

Il prodotto di due elementi **a, b** (con valori 0 e 1) dell'insieme è un elemento **c** dell'insieme (**c** cioè può avere soltanto il valore 0 oppure 1), che assume valore 1 se **a** e **b** valgono **entrambi** 1.

Nella *fig. 10* è riportata la tabella della verità del prodotto logico.

	ELEMENTI OPERANDI		ELEMENTO RISULTATO
	a	**b**	**c**
1)	0	1	0
2)	1	0	0
3)	1	1	1
4)	0	0	0

Fig. 10 – Tabella della verità dell'operazione AND (prodotto).

Si può notare che:

— nel caso 1), **c** vale 0 perché **a** vale 0 e **b** vale 1;
— nel caso 2), **c** vale 0 perché **a** vale 1 e **b** vale 0;
— nel caso 3), **c** vale 1 perché **a, b** valgono **entrambi** 1;
— nel caso 4), **c** vale 0 perché **a, b** valgono entrambi 0.

Il nome AND assegnato a questa operazione deriva anch'esso dalla lingua inglese, nella quale la congiunzione "e" viene denominata AND: infatti, come si è visto, affinché **c** sia eguale a 1 occorre che **a** AND **b** (cioè "a" e "b") siano congiuntamente 1.

Il segno grafico del circuito elettronico che svolge la funzione AND (a due ingressi, **a** e **b**) con il simbolo della rispettiva funzione logica è riportato nella *fig. 11*.

Fig. 11 – Segno grafico del circuito elettronico che svolge la funzione AND.

Si può anche dire che il circuito AND attribuisce il valore 1 a **c** se **a** e **b** valgono entrambi 1.

INVERSIONE (OPERAZIONE NOT)

Si definisce elemento inverso di **a** e si indica con $\bar{a}$ un elemento dell'insieme (a due soli valori, 0 e 1) che vale 1 se **a** vale 0 e vale 0 se **a** vale 1.

	ELEMENTO OPERANDO	ELEMENTO RISULTATO
	a	$\bar{a}$
1)	0	1
2)	1	0

Fig. 12 – Tabella della verità dell'operazione NOT (inversione).

La tavola della verità dell'inversione è riportata nella *fig. 12*: da essa risulta che:

— nel caso 1), $\bar{a}$ vale 1 perché **a** vale 0;
— nel caso 2), $\bar{a}$ vale 0 perché **a** vale 1;

Come si può constatare, l'operazione NOT trasforma il valore 1 nel valore 0 e viceversa, ossia capovolge il valore di una variabile binaria.

Come simbolo per indicare tale operazione viene usata una **lineetta**, che si traccia sulla variabile che si intende negare ($\bar{a}$).

Il segno grafico del circuito elettronico che svolge la funzione NOT con il simbolo della rispettiva funzione logica è riportato nella *fig. 13*.

Fig. 13 – Segno grafico del circuito elettronico che svolge la funzione NOT.

I circuiti elettronici OR, AND, NOT ora visti, detti **digitali**, diventano anche logici quando si assegna un significato ai valori 1 e 0 con cui operano.

In tal caso, la commutazione da un valore all'altro equivale ad un passaggio logico.

Per esempio, alla cifra binaria 1 può essere associato il consenso (SI) per l'esecuzione di una data operazione ed alla cifra binaria (0) la mancanza di tale consenso (NO).

In questo modo i valori binari possono essere utilizzati per esprimere in forma simbolica i due termini dell'alternativa.

5.1 – Proprietà fondamentali dell'algebra booleana

Nell'algebra booleana vigono alcune proprietà che hanno il carattere di postulati.

Proprietà 1

Nell'insieme dell'algebra booleana esistono due elementi unitari (0 e 1) tali che, denominato **a** un qualsiasi altro elemento dell'insieme, si ha:

$$a+0=0+a=a \qquad a\cdot 1=1\cdot a=a$$

$$a\cdot 0=0\cdot a=0 \qquad a+1=1+a=1$$

cioè, ogni elemento **a** dell'insieme, combinato (sia a destra sia a sinistra) con l'elemento unitario 0 secondo l'operazione di addizione fornisce sé stesso (cioè **a**), mentre combinato secondo l'operazione di prodotto fornisce l'elemento unitario zero.

Le seconde eguaglianze, cioè:

$$a \cdot 1 = 1 \cdot a = a$$
$$a + 1 = 1 + a = 1$$

sono ottenute per DUALITA' dalle prime.

Il **principio di dualità** è un'ulteriore proprietà dell'algebra booleana e consiste nel trasformare contemporaneamente l'operatore (+) di una relazione o proprietà booleana con l'operatore (·), e contemporaneamente l'elemento 0 con l'elemento 1.

Dopo lo scambio la proprietà che ne risulta è "vera", e tale verità si può dimostrare con un teorema (detto appunto di dualità).

Proprietà 2

Le operazioni di addizione e prodotto godono della proprietà commutativa:

addizione	$a+b=b+a$
prodotto	$a \cdot b = b \cdot a$

per ogni elemento **a, b** dell'insieme.

Proprietà 3

Le operazioni di addizione e prodotto godono della proprietà distributiva:

1) $a(b+c)=ab+ac$

2) $a+bc=(a+b)(a+c)$

Nel caso 1) il prodotto di **a** per **(b+c)** si "distribuisce" nella somma dei due prodotti **ab** ed **ac**.

Nel caso 2) la somma di **a** con **bc** si "distribuisce" nel prodotto delle due somme **(a+b)** ed **(a+c)**.

Proprietà 4

Per ogni elemento **a** dell'algebra booleana si ha:

$$a+\bar{a}=1$$
$$a\cdot\bar{a}=0$$

Anche in questo caso la seconda eguaglianza si può ottenere dalla prima per dualità.

Variabili e funzioni booleane

Si definisce **variabile booleana** un elemento qualsiasi dell'algebra di Boole, che può assumere l'uno o l'altro dei due valori binari 0 e 1; essa si indica di solito con le ultime lettere dell'alfabeto: x, y, z per distinguerla dalle costanti booleane a cui vengono assegnate le prime lettere dell'alfabeto: **a, b, c**.

Si definisce **funzione booleana** una qualsiasi combinazione di variabili e costanti booleane; le possibili combinazioni sono sempre quelle che consentono di associare tra loro due o piú elementi dell'insieme, cioè le tre operazioni logiche già viste:

$$AND = a\cdot b$$
$$OR = a+b$$
$$NOT = \bar{a}$$

Esempi di funzioni booleane sono:

$$f_1(x,y)=ab+cx+\overline{xy}$$
$$f_2(x,y)=\bar{x}+xy+x\ (\bar{x}+y)$$

Come si vede, le funzioni booleane si esprimono come le normali funzioni algebriche matematiche. Il simbolo **f** sta per "funzione" (cioè espressione di dipendenza); i nomi **x, y** che appaiono tra parentesi, a destra del simbolo di funzione **f**, rappresentano le variabili booleane indipendenti, da cui dipende il valore dell'espressione booleana che sta alla sinistra del segno di eguaglianza (=).

Quindi, supposto che le costanti siano:

$$a=0$$
$$b=1$$
$$c=1$$

calcolando le funzioni f_1 ed f_2 in base ai valori indipendenti delle variabili x, y si ottiene la tabella della *fig. 14*.

VARIABILI		FUNZIONI	
x	y	$f_1(x,y)$	$f_2(x,y)$
0	1	1	1
1	0	1	0
1	1	1	1
0	0	1	1

Fig. 14 – Tabella funzionale per il calcolo delle funzioni $f_1(x,y)$, $f_2(x,y)$ per $a=0$, $b=c=1$.

Verifichiamo ora alcuni di questi valori.

Per x=0 y=1 si ha (1ª riga della tabella della *fig. 14*):

$$f_1(0,1)=0\cdot1+1\cdot0+\overline{0\cdot1}=0+0+\bar{0}=1;$$

$$f_2(0,1)=\bar{0}+0\cdot1+0\cdot(\bar{0}+1)=1+0+0\cdot(1+1)=1+0\cdot1=1.$$

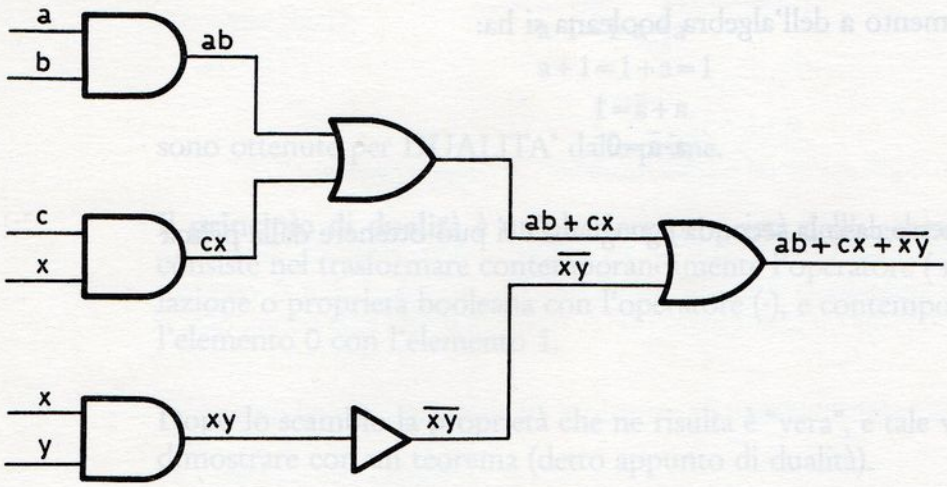


Fig. 15 – Circuito booleano corrispondente alla funzione $f_1(x,y)$.

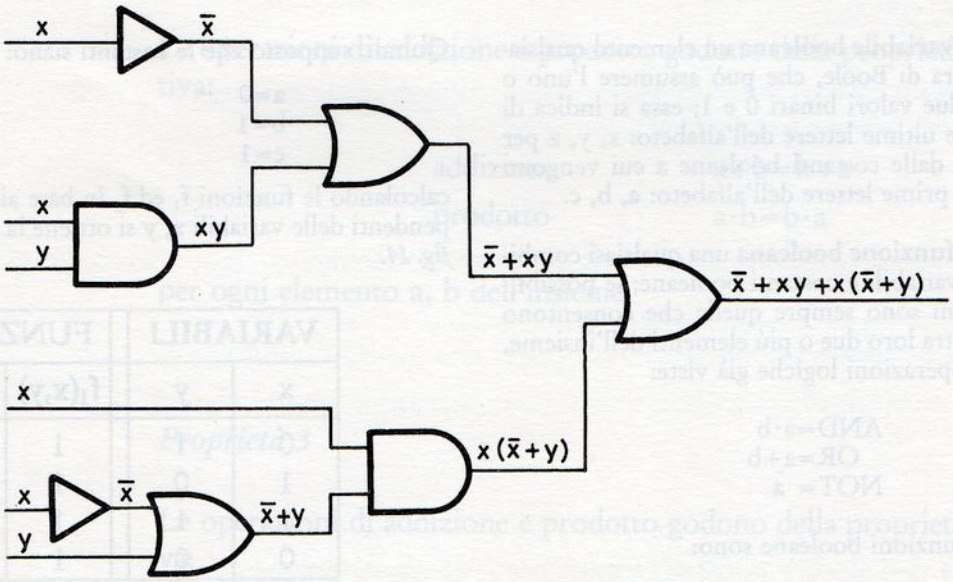


Fig. 16 – Circuito booleano corrispondente alla funzione $f_2(x,y)$.

Per x=1 e y=0 si ha (2ª riga della stessa tabella):

$f_1(1,0)=0\cdot1+1\cdot1+\overline{1\cdot0}=0+1+\overline{0}=1+1=1;$

$f_2(1,0)=\overline{1}+1\cdot0+1\cdot(\overline{1}+0)=0+0+1\cdot(0+0)=$
$=0+0+1\cdot0=0+0+0=0;$

e cosí via di seguito.

Ad ogni funzione booleana corrisponde un ben determinato circuito booleano e questo si costruisce utilizzando gli stessi simboli usati per rappresentare le tre operazioni dell'algebra booleana: AND, OR, NOT.

Per esempio, alla funzione $f_1(x,y)=ab+cx+\overline{xy}$ prima considerata, corrisponde il circuito boo-

leano riportato nella *fig. 15* ed alla funzione $f_2(x,y)=\bar{x}+xy+x(\bar{x}+y)$ corrisponde il circuito rappresentato nella *fig. 16*.

Le funzioni booleane, e corrispondentemente i circuiti booleani, si possono semplificare come le funzioni matematiche.

Se si osserva la funzione $\mathbf{f_1}$, per esempio, il prodotto (**a·b**) tra due costanti può essere sostituito da un'altra costante **a'**. Si può cioè prefabbricare a parte la nuova costante **a'**, in modo che nel circuito $\mathbf{f_1}$ fluisca soltanto l'indispensabile; anche i poli di entrata **x** possono essere unificati con un solo polo.

Per la funzione $f_2(x,y)$ si può notare che l'espressione:

$$f_2(x,y)=\bar{x}+xy+x(\bar{x}+y)$$

può semplificarsi nella seguente:

$$\begin{aligned} f'_2(x,y)&=\bar{x}+x(y+\bar{x}+y)=\\ &=\bar{x}+x(\bar{x}+y)=\bar{x}+x\bar{x}+xy=\\ &=\bar{x}(1+x)+xy=\bar{x}+xy \end{aligned}$$

Si ricordi che per la ***Proprietà 1*** si ha per ogni x dell'algebra booleana: $1+x=x+1=1$ e per la ***Proprietà 4***:

$$\begin{aligned} \bar{x}+x&=1\\ \bar{x}x&=0 \end{aligned}$$

$$\begin{aligned} \text{Inoltre: } y+y&=y+y\cdot 1\\ &=y+y(y+\bar{y})\\ &=y+yy+y\bar{y}\\ &=y(1+y)+0\\ &=y\cdot 1\\ &=y \end{aligned}$$

Il circuito corrispondente a $f'_2(x,y)$ assume quindi l'aspetto riportato nella *fig. 17*.

Tale circuito risulta equivalente a quello della *fig. 16*, ma sensibilmente piú semplice.

A riprova che il circuito booleano $f'_2(x,y)$ è equivalente a quello $f_2(x,y)$ basta sostituire i possibili valori binari alle variabili indipendenti **x,y** per ottenere la stessa tabella vista prima (colonna 4 della *fig. 14*).

Si è visto che una funzione booleana può trasformarsi in un'altra funzione equivalente.

L'equivalenza è una proprietà analoga a quella esistente in matematica quando si confrontano due funzioni qualsiasi.

Per esempio, le due funzioni lineari:

$$\begin{aligned} f_1(x)&=3x-x\\ f_2(x,z)&=23x-20x-x+z-z \end{aligned}$$

sembrano apparentemente diverse, ma se si sostituisce alle variabili indipendenti qualsiasi valore reale, si ottiene sempre:

$$f_1(x)=f_2(x,z).$$

In effetti, se si osservano un po' meglio le due precedenti funzioni, si noterà che esse si possono piú semplicemente scrivere:

$$\begin{aligned} f_1(x)&=2x\\ f_2(x,z)&=2x \end{aligned}$$

cioè le funzioni $f_1(x)=f_2(x,z)$ risultano identiche anche nella forma.

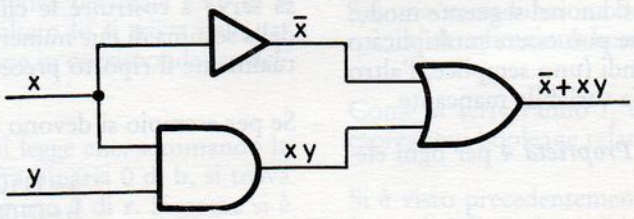


Fig. 17 – Circuito equivalente a quello della funzione $f'_2(x,y)$ riportato nella fig. 16.

Perché allora, per stabilire se due funzioni sono equivalenti, si sostituiscono i valori possibili delle variabili indipendenti alle relative espressioni?

La ragione è semplice e si potrebbe rispondere: "perché l'apparenza inganna".

Cioè due funzioni potrebbero a prima vista sembrare diverse: la sostituzione di tutti i possibili valori delle variabili nelle espressioni funzionali costituisce una prova della loro uguaglianza o disuguaglianza.

CIRCUITO ADDIZIONATORE SERIALE

Enunciamo ora un teorema fondamentale che, una volta capito, consente di abbozzare la costruzione logica del circuito addizionatore di un computer (Serial Adder).

Teorema fondamentale

Una funzione booleana, non contenente costanti, può essere scritta in una forma algebrica standard, denominata **forma disgiuntiva normale** (f.d.n.)

La forma disgiuntiva normale di una funzione booleana è un'espressione algebrica in cui le variabili (solo le variabili, perché si suppone che le costanti non esistano) sono combinate tra loro in somme di prodotti; ogni prodotto contiene come fattori "tutte" le possibili variabili della funzione, con una "sola" ricorrenza per ogni singola variabile, nella forma semplice o complementata, cioè inversa.

Con un esempio si può chiarire questo fondamentale concetto. Sia:

$$f(x,y,z)=xy+\bar{x}+\bar{z}$$

una funzione booleana nelle tre variabili indipendenti **x, y, z**, non contenente costanti booleane.

Essa si può trasformare in f.d.n. nel seguente modo: ogni termine della funzione può essere moltiplicato per la somma di due addendi (uno semplice, l'altro inverso) corrispondenti alla variabile mancante.

Si ricordi che in base alla ***Proprietà 4*** per ogni elemento x si ha:

$$x+\bar{x}=1$$
$$x\bar{x}=0$$

Quindi, se si moltiplica (a destra o a sinistra) un termine qualsiasi di una funzione booleana, ad esempio yz, per $(x+\bar{x})=1$, il valore originario di yz non viene alterato.

Nel nostro caso, il termine xy della funzione $f(x,y,z)=xy+\bar{x}+\bar{z}$ si può moltiplicare per $(z+\bar{z})$; il termine $\bar{x}$ per $(y+\bar{y})(z+\bar{z})$; il termine $\bar{z}$ per $(x+\bar{x})(y+\bar{y})$.

Quindi si ha:

$$f(x,y,z)=xy(z+\bar{z})+\bar{x}(y+\bar{y})(z+\bar{z})+\bar{z}(x+\bar{x})(y+\bar{y})$$

Sviluppando i singoli prodotti si ottiene:

$$\begin{aligned}f(x,y,z)&=xyz+xy\bar{z}+\bar{x}(yz+y\bar{z}+\bar{y}z+\bar{y}\bar{z})+\\&+\bar{z}(xy+x\bar{y}+\bar{x}y+\bar{x}\bar{y})=\\&=xyz+xy\bar{z}+\bar{x}yz+\bar{x}y\bar{z}+\bar{x}\bar{y}z+\\&+\bar{x}\bar{y}\bar{z}+\bar{z}xy+\bar{z}x\bar{y}+\bar{z}\bar{x}y+\bar{z}\bar{x}\bar{y}.\end{aligned}$$

Ora, ricordando alcune proprietà dell'algebra booleana (come per esempio **a+a=a**), l'espressione precedente si può semplificare ancora nella seguente:

$$\begin{aligned}f'(x,y,z)&=xyz+xy\bar{z}+\bar{x}yz+\bar{x}y\bar{z}+\bar{x}\bar{y}z+\\&\bar{x}\bar{y}\bar{z}+\bar{z}xy+\bar{z}x\bar{y}+\bar{z}\bar{x}y\end{aligned}$$

la quale ha un termine in meno della precedente.

Questa è la forma disgiuntiva normale della funzione originaria:

$$f(x,y,z)=xy+\bar{x}+\bar{z}$$

Si può verificare che $f'(x,y,z)=f(x,y,z)$.

Si osservi ora la tabella funzionale di addizione tra due addendi **a, b,** binari, riportata nella *fig. 18*; essa serve a costruire le cifre binarie del risultato S della somma di due numeri binari, utilizzando eventualmente il riporto precedente **rp**.

Se per esempio si devono addizionare le cifre:

$$a=111001$$
$$b=010010$$

la somma S=a+b eseguita con le normali regole arit-

	1	2	3	4	5	6	7	8
a	0	0	1	1	0	0	1	1
b	0	1	0	1	0	1	0	1
r_p	0	0	0	0	1	1	1	1
S	0	1	1	0	1	0	0	1
r	0	0	0	1	0	1	1	1

*Fig. 18 - Tavola funzionale di addizione fra due numeri binari **a, b**.*

metiche del sistema binario (base=2) risulta:

$$\begin{array}{r} 111001+ \\ 010010= \\ \hline 1001011 \end{array}$$

Si può anche eseguire questa somma utilizzando la tabella funzionale della *fig. 18*, dove in corrispondenza delle righe intestate **a, b** ci sono i possibili valori binari di questi due operandi; in corrispondenza della riga **rp** i valori binari del riporto precedente (cioè il riporto di una somma binaria effettuata in una fase precedente); in corrispondenza della riga S i valori binari di una somma binaria effettuata nella fase corrente; in corrispondenza di **r** i valori binari del riporto corrente.

Ritorniamo ora alla somma S=a+b.

Come nel sistema decimale, la somma tra le cifre dei due addendi si inizia sempre da destra (cioè dai bit 0,1 meno significativi); in questa posizione dobbiamo eseguire:

$$\begin{array}{r} 1+ \\ 0 \end{array}$$

Secondo l'aritmetica booleana 1+0=1 e riporto 0 (come d'altronde in aritmetica decimale 9+0 fa 9 con riporto 0); questo risultato lo si trova nella tabella funzionale di addizione in corrispondenza della colonna 3.

In questa colonna infatti si legge che, sommando la cifra binaria 1 di **a** alla cifra binaria 0 di **b**, si trova la cifra binaria 1 di S e riporto 0 di **r**. Siccome si è incominciato ad operare a destra, nella prima operazione il riporto precedente **rp** è necessariamente 0.

Proseguendo nella somma di a+b si ha:

per la 2ª cifra binaria di S: 0+1=1 e riporto 0
per la 3ª cifra binaria di S: 0+0=0 e riporto 0
per la 4ª cifra binaria di S: 1+0=1 e riporto 0
per la 5ª cifra binaria di S: 1+1=0 e riporto 1
per la 6ª cifra binaria di S: 1+0+1(rip.)=0 e riporto 1
per la 7ª cifra binaria di S: 0+0+1(rip.)=1 e riporto 0

Si ottiene quindi il risultato finale:

$$S=111001+010010=1001011$$

Come si vede, la somma S ed il riporto **r** si costruiscono (cioè dipendono) dai valori binari degli elementi **a, b, rp**.

Come si fa ad esprimere questa dipendenza in maniera analitica (cioè in formule)?

Occorre scrivere che le cifre binarie di S e di **r** sono funzioni delle cifre binarie di **a, b, rp**.

Ora se si indica con a_i, b_i, r_{pi-1}, s_i, r_i la generica cifra binaria corrispondente alla fase **i**^{ma} dell'addizione, si possono scrivere le due funzioni analitiche booleane:

$$S_i=f_1(a_i, b_i, r_{pi-1})$$
$$r_i=f_2(a_i, b_i, r_{pi-1})$$

che si leggono in questo modo: le cifre binarie S_i sono dipendenti, in un certo modo (cioè secondo la relazione di dipendenza f_1), dalle cifre binarie a_i, b_i, r_{pi-1}; le cifre binarie r_i sono dipendenti, in un altro modo (cioè secondo la relazione di dipendenza f_2), dalle stesse cifre binarie a_i, b_i, r_{pi-1}.

Piú sinteticamente, si può anche dire:

S_i è una certa funzione booleana f_1 di a_i, b_i, r_{pi-1};
r_i è un'altra funzione booleana f_2 di a_i, b_i, r_{pi-1}.

Come si scriveranno f_1 e f_2, cioè quali saranno le espressioni booleane relative a f_1 e f_2?

Si è visto precedentemente che ogni funzione booleana si può esprimere in forma disgiuntiva normale (f.d.n.).

Iniziamo allora con la funzione:

$$S_i = f_1 (a_i, b_i, r_{pi-1}).$$

Come si è visto, una funzione booleana scritta in f.d.n. è una somma di prodotti fondamentali (tali cioè da contenere, in forma semplice o complementata, tutte le variabili indipendenti, con un sola ricorrenza per volta).

La funzione S_i in f.d.n. si scrive:

$$S_i = a_i \bar{b}_i \bar{r}_{pi-1} + \bar{a}_i b_i \bar{r}_{pi-1} + \bar{a}_i \bar{b}_i r_{pi-1} + a_i b_i r_{pi-1}$$

Come si è ottenuta questa espressione?

Occorre osservare la tabella di addizione precedente (*fig. 18*) in corrispondenza delle colonne in cui il valore binario di S_i è 1; perché S_i risulti 1, occorre che uno solo degli otto prodotti fondamentali abbia il valore 1 e questo prodotto fondamentale si scriverà quindi nella colonna in cui $S_i = 1$ (una delle colonne 2, 3, 5, 8 della tabella della *fig. 18*), la variabile indipendente (una delle tre $\mathbf{a_i}$, $\mathbf{b_i}$, $\mathbf{r_{pi-1}}$) che appare con il valore 1 sarà scritta in forma semplice, le altre che hanno il valore 0 saranno scritte in forma complementata o inversa; solo così infatti il prodotto fondamentale assume il valore 1.

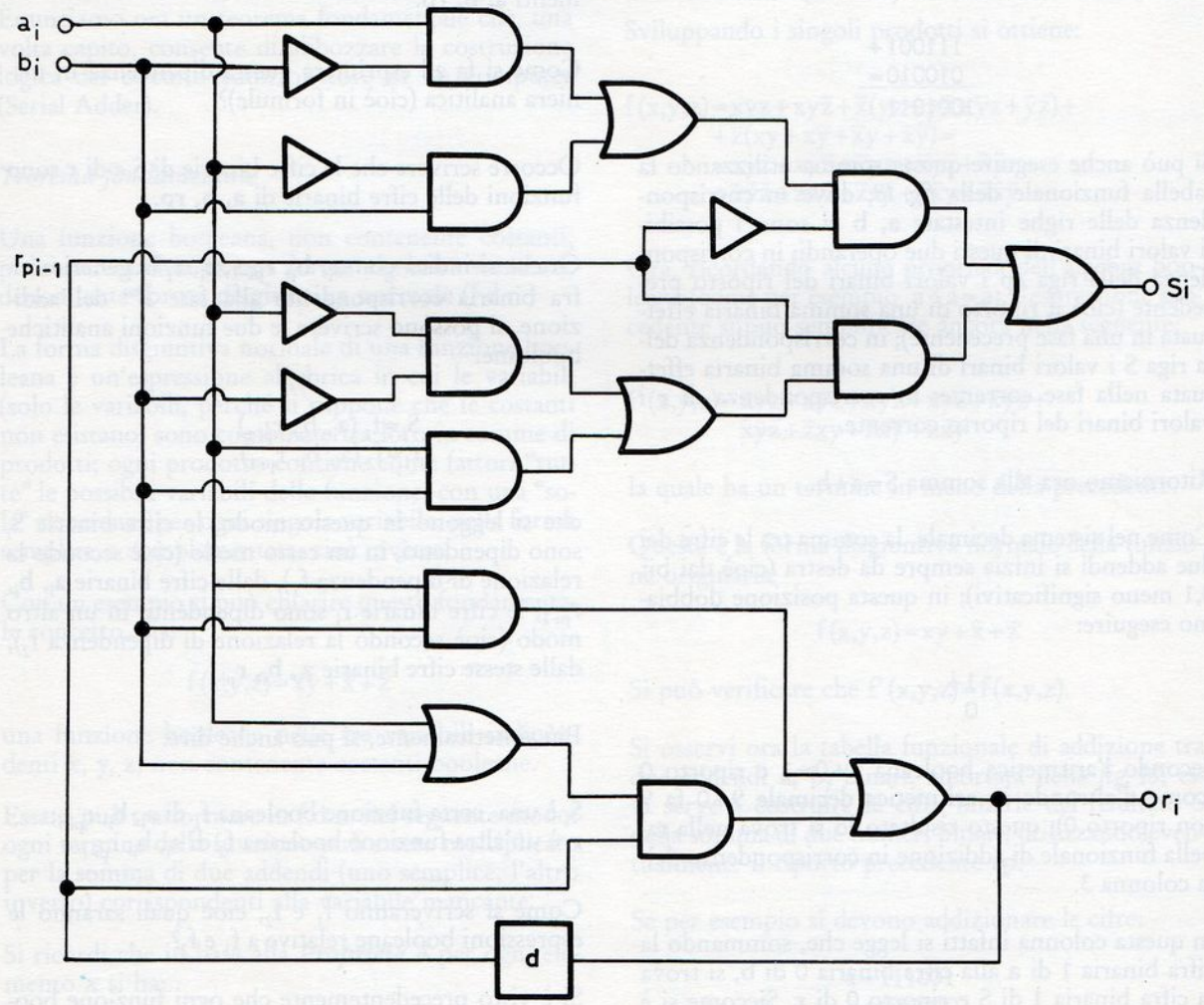


Fig. 19 – Circuito sommatore (Serial Adder Circuit).

Quindi, per la colonna 2 si ha il prodotto fondamentale $\bar{a}_i\ b_i\ \bar{r}_{pi-1}$; per la colonna 3 si ha il secondo prodotto fondamentale $a_i\ \bar{b}_i\ \bar{r}_{pi-1}$, per la colonna 5 il prodotto $\bar{a}_i\ \bar{b}_i\ r_{pi-1}$ e per la colonna 8 il quarto prodotto fondamentale $a_i\ b_i\ r_{pi-1}$.

Sommando i quattro prodotti fondamentali (come richiede il teorema), si ottiene la funzione booleana:

$$S_i = f_1(a_i, b_i, r_{pi-1}) = \bar{a}_i b_i \bar{r}_{pi-1} + a_i \bar{b}_i \bar{r}_{pi-1} + \bar{a}_i \bar{b}_i r_{pi-1} + a_i b_i r_{pi-1}$$

Analogamente, per r_i si ottiene:

$$r_i = f_2(a_i, b_i, r_{pi-1}) = a_i b_i \bar{r}_{pi-1} + \bar{a}_i b_i r_{pi-1} + a_i \bar{b}_i r_{pi-1} + a_i b_i r_{pi-1}$$

E' possibile semplificare ulteriormente S_i e r_i nel seguente modo:

$$S_i = \bar{r}_{pi-1}(a_i \bar{b}_i + \bar{a}_i b_i) + r_{pi-1}(\bar{a}_i \bar{b}_i + a_i b_i);$$

$$r_i = (a_i b_i\ (\bar{r}_{pi-1} + r_{pi-1}) + r_{pi-1}(\bar{a}_i b_i + a_i \bar{b}_i) =$$

$$= a_i b_i + r_{pi-1}(\bar{a}_i b_i + \bar{a}_i \bar{b}_i).$$

A questo punto possiamo tracciare il circuito addizionatore come mostrato nella *fig. 19.*

Il quadratino d rappresenta un elemento di ritardo che viene applicato allo stadio i^{mo} dell'addizione, al riporto r_{pi-1} generato dallo stadio precedente.

Da memorizzare

- Ogni sistema di numerazione ha una base; la base del sistema decimale è il numero 10, la base del sistema binario è il numero 2, ecc. La base si può quindi considerare il **numero simbolico** del sistema da cui riparte la numerazione.
- L'individuazione della posizione di una cifra in un certo sistema di numerazione è utile per calcolare il corrispondente valore reale del numero (sinonimo di valore decimale).
- Oltre ai sistemi di numerazione decimale e binario i computer utilizzano anche altri sistemi che sono deducibili quasi tutti dal sistema binario, come i sistemi di numerazione **ottale** ed **esadecimale**.
- Un numero binario si può trasformare nel sistema ottale raggruppando a 3 a 3 le cifre binarie a partire da destra.
- Un numero binario si può trasformare nel sistema esadecimale raggruppando a 4 a 4 le cifre binarie a partire da destra.
- Nell'ambito della scienza dei calcolatori, la variabile è un'entità numerica o alfanumerica che nel corso dell'elaborazione può assumere un qualsiasi valore indipendente dal valore di altre variabili. La funzione invece è una entità, sempre numerica o alfanumerica, il cui valore dipende (è funzione di....) da una o piú altre variabili indipendenti.
- Le quattro funzioni trigonometriche senx, cosx, tgx e cotgx sono calcolabili su un computer mediante l'utilizzo di speciali nomi che nel linguaggio BASIC sono: SIN(X), COS(X), TAN(X) e ATN(X). Da queste funzioni principali possono essere derivate altre funzioni dette appunto funzioni derivate.
- La nomenclatura usata nel linguaggio BASIC per le funzioni logaritmiche ed esponenziali è: LOG(X); EXP(X). Anche da queste funzioni ne possono essere derivate molte altre.
- Si definisce algebra booleana un insieme finito di elementi **a**, **b**, **c** che possono avere due soli valori: 0, 1, ed il complesso delle operazioni con cui sono combinati tra loro.
- Le operazioni dell'algebra booleana, o delle variabili binarie, sono tre: addizione (OR), prodotto (AND) ed inversione (NOT).
- La somma (OR) di due elementi **a**, **b** dell'insieme è un elemento **c** dell'insieme che assume valore 1 se **almeno** uno dei valori di **a** oppure di $\overline{b}$ è di valore 1.
- Il prodotto (AND) di due elementi **a**, **b** dell'insieme è un elemento **c** dell'insieme che vale 1 se **a** e **b** valgono **entrambi** 1.
- Si definisce elemento inverso di **a**, e si indica con $\overline{a}$, un elemento dell'insieme che vale 1 se **a** vale 0 e vale 0 se **a** vale 1.
- Le operazioni di somma e prodotto dell'algebra di Boole corrispondono alle analoghe operazioni dell'aritmetica ordinaria, mentre la inversione o negazione non ha corrispondenze.
- Le operazioni di addizione e prodotto dell'algebra booleana godono delle proprietà commutativa e distributiva.
- Per ogni elemento dell'algebra booleana si ha: $a+\overline{a}=1$; $a\cdot\overline{a}=0$

In seguito...

Nella prossima lezione verrà approfondito l'uso delle subroutine in BASIC, prendendo in considerazione alcuni esempi pratici che ne evidenziano le caratteristiche piú salienti.

Verrà poi esaminato l'impiego del comando **DEF FN**, che consente di definire con un breve nome una funzione matematica, e dell'istruzione **RANDOMIZE**, la quale permette di variare la sequenza dei numeri casuali generati dalla funzione RND.

Scuola Radio Elettra
Torino

Pratica del BASIC

Lezione 9

1 — Matematica con il computer
2 — Forma abbreviata di alcune istruzioni
3 — Grafica sul computer C64
4 — Grafica sul computer VIC-20

Che cosa farà

In questa lezione avrà modo di esercitarsi nel calcolo di alcune funzioni matematiche con il computer.

Quindi verranno esaminate le abbreviazioni delle parole chiavi del linguaggio BASIC, che risultano particolarmente utili quando si deve battere un programma piuttosto lungo.

Infine, nell'ultima parte della lezione, che è dedicata alla grafica con il computer, apprenderà il modo in cui si procede per comporre sullo schermo disegni utilizzando le istruzioni **READ** e **DATA** e per tracciare linee mediante cicli **FOR... NEXT....**

1. - Matematica con il computer

L'utilizzo del calcolo numerico con il computer, visto nella lezione *Analisi e programmazione 9*, ha comportato un primo approccio, anche se necessariamente non completo, con l'analisi numerica.

In questa lezione Lei si eserciterà nel calcolo di alcune funzioni matematiche, servendosi del computer in Suo possesso.

ESERCIZIO 1

Iniziamo con il seguente, semplice programma BASIC, nel quale sono programmate le principali funzioni trigonometriche:

$$\text{SIN}(X),\ \text{COS}(X),\ \text{TAN}(X),$$

dove X rappresenta in radianti un angolo compreso tra 0° e 360° (angolo giro).

Il programma prevede la stampa (visualizzazione) su schermo di otto valori delle funzioni trigonometriche corrispondenti agli otto valori (argomenti) angolari:

$$X = 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}.$$

Come sappiamo, questi sono angoli espressi in radianti e corrispondono rispettivamente agli angoli sessagesimali:

$$X° = 0,\ 45°,\ 90°,\ 135°,\ 180°,\ 225°,\ 270°,\ 315°.$$

Per eseguire i calcoli delle funzioni trigonometriche viene utilizzata l'istruzione ciclica BASIC che segue:

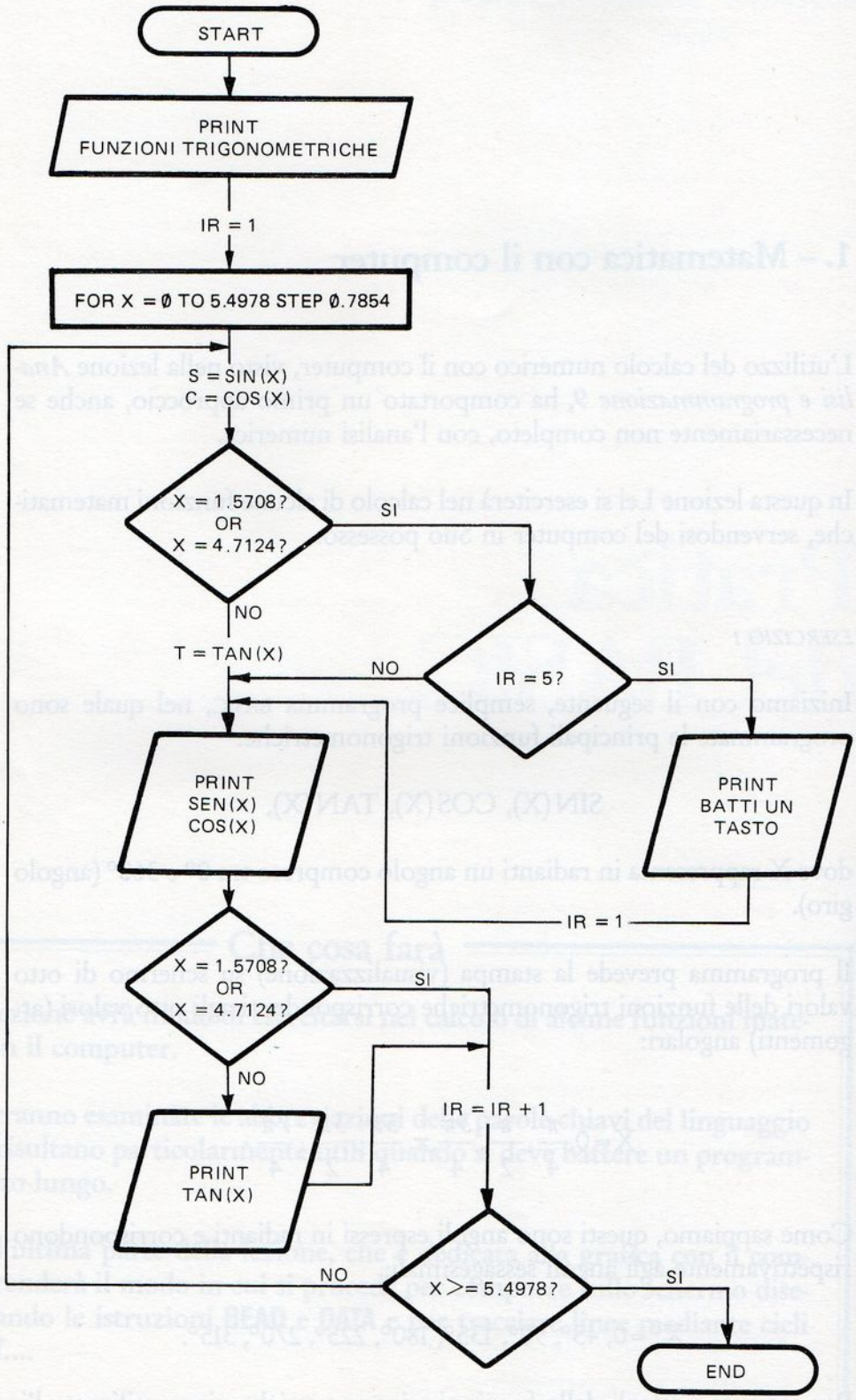


Fig. 1 – Flow-chart del programma che calcola le funzioni SIN(X), COS(X), TAN(X).

FOR X=Ø TO 5.4978 **STEP** Ø.7854

angolo 315° ——— angolo 45°

dove i valori 5.4978 e Ø.7854 sono rispettivamente l'angolo finale del ciclo corrispondente a 315° ed il passo o incremento angolare corrispondente a 45°.

Il flow-chart del programma in questione ed il relativo listato in linguaggio BASIC sono riportati rispettivamente nella *fig. 1* e nella *fig. 2*.

L'istruzione 4Ø provvede ad inizializzare il conteggio del numero dei gruppi di funzioni trigonometriche da visualizzare sullo schermo, i quali devono essere non piú di quattro per volta (istruzione 7Ø): lo schermo del C64, come è noto, può contenere al massimo venticinque righe.

Il ciclo di programma costituito dalle istruzioni 45 e 11Ø provvede a calcolare ed a visualizzare sullo schermo i valori delle tre funzioni trigo-

```
10 REM********************
15 REM*CALCOLO FUNZIONI*
20 REM*  SEN,COS,TAN   *
25 REM********************
30 PRINT"□"
35 PRINT"FUNZIONI TRIGONOMETRICHE"
40 IR=1
45 FOR X=Ø TO 5.4978 STEP Ø.7854
50 S=SIN(X)
55 C=COS(X)
60 IF X=1.5708 OR X=4.7124 THEN 70
65 T=TAN(X)
70 IF IR=5 THEN 115
75 PRINT"-----------------------------------------"
80 PRINT "PER X=";X
85 PRINT "SEN(X)=";S
90 PRINT "COS(X)=";C
95 IF X=1.5708 OR X=4.7124 THEN 105
100 PRINT "TAN(X)=";T
105 IR=IR+1
107 IF X>=5.4978 THEN 125
110 NEXT X
115 PRINT"BATTI UN TASTO"
117 GET L$
119 IF L$="" THEN 117
120 PRINT"□";:IR=1:GOTO 75
125 END
```

Fig. 2 – Codifica in linguaggio BASIC del flow-chart della fig. 1 (per C64).

nometriche per valori di X compresi fra Ø e 5.4978, con incrementi di Ø.7854.

Come può constatare dal programma listato nella *fig. 2*, sono previste due diramazioni condizionate (istruzione 6Ø ed istruzione 95) nel caso in cui si debba calcolare la tang ($\pi/2$), corrispondente a 1.5708, oppure (OR) la tang ($3\pi/2$), corrispondente a 4.7124, perché, come è noto, tali valori delle tangenti sono infiniti; piú esattamente, $+\infty$ per tang ($\pi/2$) e $-\infty$ per tang ($3\pi/2$), essendo ∞ il simbolo generalmente usato in matematica per indicare una quantità grandissima.

Quando X assume il valore 5.4978 (istruzione 1Ø7), l'elaborazione termina definitivamente (istruzione 125).

Ad ogni videata (visualizzazione dei quattro gruppi di funzioni trigonometriche) compare il messaggio:

BATTI UN TASTO

(istruzione 115) e l'elaborazione si ferma per effetto dell'iterazione costituita dalle istruzioni 117 e 119.

Desiderando procedere nella visualizzazione, è sufficiente, come è noto, battere un tasto qualsiasi della tastiera.

Se nel programma in esame si vuole raddoppiare il calcolo delle funzioni trigonometriche, basta dimezzare il passo di elaborazione riportato nella istruzione 45 (passo che, come già precisato, corrisponde a 45 gradi od a $\pi/4$ radianti).

Si tratta di sostituire il valore $\pi/4$=Ø.7854 dell'istruzione 45 con la sua metà, cioè $\pi/8$=Ø.3927.

Si può ancora fare notare che l'istruzione 75 provvede a stampare **38 trattini orizzontali**, aventi lo scopo di separare fra loro ciascun gruppo di funzioni trigonometriche.

Deve quindi fare attenzione a non battere un numero maggiore di trattini, per non determinare una interlinea fra i trattini stessi ed ogni gruppo di funzioni calcolate.

Se utilizza il VIC-20 deve modificare tale istruzione, a causa della minore larghezza dello schermo, battendo venti trattini orizzontali anziché trentotto.

Per lo stesso motivo dovrà modificare l'istruzione 35 come indicato di seguito:

```
:
35 PRINT "FUNZ. TRIGONOMETRICHE"
:
```

Infine, poiché lo schermo del VIC-20 ha un minore numero di righe rispetto al C64, dovrà pure modificare l'istruzione 7Ø come segue:

```
:
7Ø IF IR=4 THEN 115
:
```

Con il VIC-20 saranno quindi visualizzati tre gruppi di funzioni trigonometriche per ogni videata.

La descrizione del listato della *fig. 2* è cosí terminata. Può senz'altro immetterlo nel computer ed eseguirlo con il comando **RUN**. Sullo schermo compariranno i previsti valori di SEN (X), COS (X) e TAN (X), come di seguito indicato:

```
FUNZIONI TRIGONOMETRICHE
PER X=  Ø
SEN (X)=  Ø
COS (X)=  1
TAN (X)=  Ø
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=  .7854
SEN (X)=  .7Ø71Ø8Ø8
COS (X)=  .7Ø7Ø5482
TAN (X)=  1.ØØØØØ367
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=  1.57Ø8
SEN (X)=  1
COS (X)=-3.6733873 E-Ø6
TAN (X)=-272228.3Ø6
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=  2.3562
SEN (X)=  .7Ø7Ø2885
COS (X)=-.7Ø7Ø11Ø78
TAN (X)=-.99998898

BATTI UN TASTO
```

Volendo visualizzare un altro gruppo di funzioni trigonometriche, è sufficiente battere un tasto qualsiasi.

Se osserva i risultati ottenuti, può constatare che i valori delle prime due funzioni trigonometriche variano in modo continuo tra i valori –1 e +1, mentre i valori della tang (x) variano tra valori positivi e valori negativi in modo discontinuo: per valori minori e prossimi a $\pi/2$ o $3\pi/2$ la tangente assume valori positivi grandissimi, mentre per valori maggiori e prossimi a $\pi/2$ o $3\pi/2$ essa assume valori negativi grandissimi.

ESERCIZIO 2

Possiamo ora scrivere un altro programma che calcola le funzioni logaritmiche ed esponenziali per certi valori della variabile indipendente x.

Tali valori non possono essere negativi per la funzione logaritmica, mentre possono essere costituiti da qualsiasi valore reale (positivo o negativo) per la funzione esponenziale.

Infatti la funzione:

$$y = \log_e(x)$$

con x negativo non esiste, perché non esiste nessun numero reale y tale per cui:

$$e^y = x,$$

cioè tale per cui e^y sia negativo.

Invece, la funzione e^x esiste per valori negativi e positivi di x.

Per valori di x negativi molto grandi, e^x (cioè $2{,}78^x$) assume valori prossimi a zero e per valori crescenti di x assume sempre valori positivi, che crescono in modo cosiddetto esponenziale.

Nella lezione *Analisi e programmazione 9* sono state riportate le rappresentazioni geometriche della funzione logaritmica (in base e=2,781) e della funzione esponenziale.

Nella *fig. 3* e nella *fig. 4* sono rappresentati rispettivamente il flow-chart e

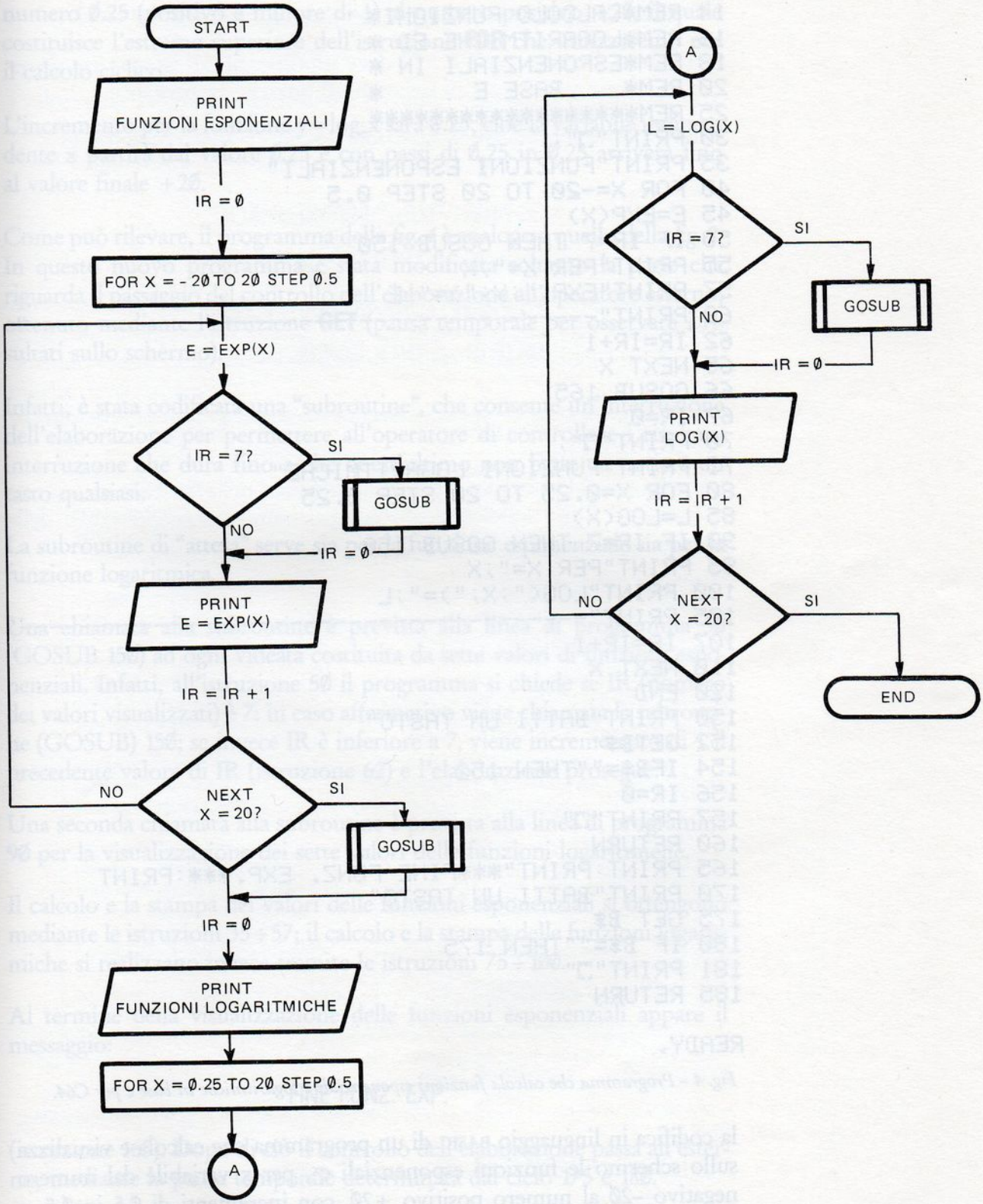


Fig. 3 – Flow-chart del programma che calcola funzioni esponenziali e logaritmiche in base e.

```
 10 REM*******************
 14 REM*CALCOLO FUNZIONI*
 16 REM*LOGARITMICHE ED *
 18 REM*ESPONENZIALI IN *
 20 REM*     BASE E      *
 25 REM*******************
 30 PRINT"□";
 35 PRINT"FUNZIONI ESPONENZIALI"
 40 FOR X=-20 TO 20 STEP 0.5
 45 E=EXP(X)
 50 IF IR=7 THEN GOSUB 150
 55 PRINT"PER X=";X
 57 PRINT"EXP(";X;")=";E
 60 PRINT"----------------------------------------"
 62 IR=IR+1
 65 NEXT X
 66 GOSUB 165
 67 IR=0
 70 PRINT"□";
 75 PRINT"FUNZIONI LOGARITMICHE"
 80 FOR X=0.25 TO 20 STEP 0.25
 85 L=LOG(X)
 90 IF IR=7 THEN GOSUB 150
 95 PRINT"PER X=";X
100 PRINT"LOG(";X;")=";L
105 PRINT"----------------------------------------"
107 IR=IR+1
110 NEXT X
120 END
150 PRINT"BATTI UN TASTO"
152 GETB$
154 IFB$=""THEN 152
156 IR=0
157 PRINT"□"
160 RETURN
165 PRINT:PRINT"***FINE FUNZ. EXP.***:PRINT
170 PRINT"BATTI UN TASTO"
175 GET B$
180 IF B$=""THEN 175
181 PRINT"□"
185 RETURN

READY.
```

Fig. 4 – Programma che calcola funzioni esponenziali e logaritmiche in base e per C64.

la codifica in linguaggio BASIC di un programma che calcola e visualizza sullo schermo le funzioni esponenziali e^x, per x variabile dal numero negativo –2Ø al numero positivo +2Ø, con incrementi di Ø.5 in Ø.5, e subito dopo calcola e visualizza la funzione $\log_e x$, per x variabile dal

numero Ø.25 (positivo e minore di 1) al numero positivo +2Ø, il quale costituisce l'estremo superiore dell'istruzione **FOR** che utilizzeremo per il calcolo ciclico.

L'incremento per la funzione $y = \log_e x$ sarà Ø.25, cioè la variabile indipendente x partirà dal valore Ø.25 e con passi di Ø.25 in Ø.25 arriverà fino al valore finale +2Ø.

Come può rilevare, il programma della *fig. 4* è analogo a quello della *fig. 2*. In questo nuovo programma è stata modificata soltanto la parte che riguarda il passaggio del controllo dell'elaborazione all'operatore esterno, ottenuto mediante l'istruzione **GET** (pausa temporale per osservare i risultati sullo schermo).

Infatti, è stata codificata una "subroutine", che consente un'interruzione dell'elaborazione per permettere all'operatore di controllare i risultati, interruzione che dura fino a che quest'ultimo non batte da tastiera un tasto qualsiasi.

La subroutine di "attesa" serve sia per la funzione esponenziale sia per la funzione logaritmica.

Una chiamata alla subroutine è prevista alla linea di programma 5Ø (GOSUB 15Ø) ad ogni videata costituita da sette valori di funzioni esponenziali. Infatti, all'istruzione 5Ø il programma si chiede se IR (numero dei valori visualizzati) è 7: in caso affermativo viene chiamata la subroutine (GOSUB) 15Ø; se invece IR è inferiore a 7, viene incrementato di 1 il precedente valore di IR (istruzione 62) e l'elaborazione prosegue.

Una seconda chiamata alla subroutine è prevista alla linea di programma 9Ø per la visualizzazione dei sette valori delle funzioni logaritmiche.

Il calcolo e la stampa dei valori delle funzioni esponenziali si ottengono mediante le istruzioni 35÷57; il calcolo e la stampa delle funzioni logaritmiche si realizzano invece tramite le istruzioni 75÷1ØØ.

Al termine della visualizzazione delle funzioni esponenziali appare il messaggio:

FINE FUNZ. EXP.

(istruzione 165). Dopo di ciò il controllo dell'elaborazione passa all'esterno, mediante la pausa temporale determinata dal ciclo 175 e 18Ø.

Le istruzioni 6Ø e 1Ø5 provvedono a visualizzare sullo schermo trentot-

to trattini di separazione fra ciascun valore di funzione.

Se utilizza il computer VIC-20 deve modificare tali istruzioni, riportando in esse venti trattini orizzontali anziché trentotto.

Dovrà inoltre modificare le istruzioni 35, 5Ø, 75 e 9Ø come segue:

```
:
35 PRINT "FUNZ. ESPONENZIALI"
:
50 IF IR=5 THEN GOSUB 150
:
75 PRINT "FUNZ. LOGARITMICHE"
:
90 IF IR=5 THEN GOSUB 150
:
```

Con il VIC-20 saranno quindi visualizzati cinque gruppi di funzioni per ogni videata.

Immetta ora nel computer il programma listato nella *fig. 4* e lo esegua con il comando **RUN**. Sullo schermo compariranno i seguenti valori delle funzioni esponenziali:

```
FUNZIONI ESPONENZIALI
PER X=-20
EXP (-20)=  2.06115362 E-09
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=-19.5
EXP (-19.5)=   3.39826782 E-09
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=-19
EXP (-19)=   5.60279644 E-09
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=-18.5
EXP (-18.5)=  9.23744967 E-09
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=-18
EXP (-18)=   1.52299798 E-08
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=-17.5
EXP (-17.5)=   2.51099916 E-08
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PER X=-17
EXP (-17)=   4.13993772 E-08
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
BATTI UN TASTO
```

Se desidera visualizzare gli altri valori, batta un tasto qualsiasi della tastiera.

I valori delle funzioni logaritmiche verranno calcolati e visualizzati sullo schermo al termine del calcolo delle funzioni esponenziali, quando sul video comparirà il messaggio:

```
***FINE FUNZ. EXP***
BATTI UN TASTO
```

A conclusione dell'argomento, occorre precisare che le funzioni matematiche non direttamente previste nel linguaggio BASIC del C64 e del VIC-20 possono essere facilmente calcolate seguendo le indicazioni fornite nella tabella della *fig. 5*.

FUNZIONE	EQUIVALENTE IN BASIC
SECANTE	SEC(X)=1/COS(X)
COSECANTE	CSC(X)=1/SIN(X)
COTANGENTE	COT(X)=1/TAN(X)
SENO INVERSO	ARCSIN(X)=ATN(X/SQR(−X*X+1))
COSENO INVERSO	ARCCOS(X)=−ATN(X/SQR (−X*X +1)) +π/2
SECANTE INVERSA	ARCSEC(X)=ATN(X/SQR(X*X−1))
COSECANTE INVERSA	ARCCSC(X)=ATN(X/SQR(X*X−1)) +(SGN(X)−1*π/2
COTANGENTE INVERSA	ARCOT(X)=ATN(X)+π/2
SENO IPERBOLICO	SINH(X)=(EXP(X)−EXP(−X))/2
COSENO IPERBOLICO	COSH(X)=(EXP(X)+EXP(−X))/2
TANGENTE IPERBOLICA	TANH(X)=EXP(−X)/(EXP(x)+EXP (−X))*2+1
SECANTE IPERBOLICA	SECH(X)=2/(EXP(X)+EXP(−X))
COSECANTE IPERBOLICA	CSCH(X)=2/(EXP(X)−EXP(−X))
COTANGENTE IPERBOLICA	COTH(X)=EXP(−X)/(EXP(X) −EXP(−X))*2+1
SENO IPERBOLICO INVERSO	ARCSINH(X)=LOG(X+SQR(X*X+1))
COSENO IPERBOLICO INVERSO	ARCCOSH(X)=LOG(X+SQR(X*X−1))
TANGENTE IPERBOLICA INVERSA	ARCTANH(X)=LOG((1+X)/(1−X))/2
SECANTE IPERBOLICA INVERSA	ARCSECH(X)=LOG((SQR (−X*X+1)+1/X)
COSECANTE IPERBOLICA INVERSA	ARCCSCH(X)=LOG((SGN(X)*SQR (X*X+1/x)
COTANGENTE IPERBOLICA INVERSA	ARCCOTH(X)=LOG((X+1)/(x−1))/2

Fig. 5 – Derivazioni di funzioni matematiche in linguaggio BASIC sul C64 e sul VIC-20.

ESERCIZIO 3

Nella lezione *Analisi e programmazione 9* è stata esaminata, fra l'altro, la numerazione binaria (ossia in base 2), la quale risulta particolarmente adatta al computer.

Infatti, come già precedentemente precisato, i due stati elettrici dei circuiti elettronici costituenti il computer possono essere convenientemente rappresentati con le cifre binarie 1 (acceso, funzionante, ecc.) e Ø (spento, inattivo, ecc.).

Il breve programma listato nella *fig. 6* consente appunto di convertire un numero decimale intero nel corrispondente binario: esaminiamone le fasi operative.

```
10 PRINT"J"
20 PRINT"■■■■CONVERSIONE DECIMALE-BINARIA"
30 PRINT
40 INPUT"NUMERO DECIMALE INTERO";D
60 B$=STR$(D-INT(D/2)*2)+B$
70 D=INT(D/2)
80 IF D<>0 THEN 60
85 PRINT
90 PRINT"CORRISPONDE IN BINARIO:";B$
100 END
```

Fig. 6 – Programma in linguaggio BASIC *che opera la conversione di numeri decimali interi in numeri binari.*

Il programma inizia con l'istruzione 2Ø, che prevede la stampa del messaggio:

CONVERSIONE DECIMALE-BINARIA

In questa istruzione, dopo il carattere virgolette (") di apertura della stringa, occorre battere quattro volte il tasto ⇐ CRSR ⇒ per centrare la scritta sullo schermo.

Se usa il computer VIC-20, tale istruzione deve essere cosí modificata:

```
:
20 PRINT "CONVERSIONE"
25 PRINT "DECIMALE-BINARIA"
:
```

Nell'istruzione 4Ø c'è una **INPUT** con messaggio, il quale avverte l'operatore che può introdurre il numero decimale intero (D) da convertire in binario.

Il ciclo di programma costituito dalle istruzioni 6Ø÷8Ø esegue il calcolo per ottenere la conversione richiesta.

In particolare, l'istruzione 6Ø controlla se il numero D è dispari, e per fare ciò sottrae due volte il valore intero dallo stesso numero D, ossia (D-INT (D/2)*2).

Se dal controllo eseguito il numero si rivela dispari, il risultato ad esso corrispondente è 1; se invece si rivela pari, il risultato è Ø.

Sempre a proposito dell'istruzione 6Ø, occorre notare che la funzione **STR$** (la quale verrà esaminata in una prossima lezione) consente al computer di trattare i numeri binari come una stringa alfanumerica.

L'istruzione di diramazione presente nella linea di programma 8Ø verifica se il numero D, dopo la divisione per due operata dall'istruzione 7Ø, è diverso da zero: in caso affermativo l'elaborazione ritorna all'istruzione 6Ø, da dove riprende il ciclo di calcolo.

Se invece D è uguale a zero, l'elaborazione termina ed il numero binario viene visualizzato tramite l'istruzione 9Ø.

Usando il computer VIC-20, questa istruzione deve essere così modificata:

```
⋮
9Ø PRINT "CORRISP. IN BINARIO:";B$
```

Immetta nel computer il programma listato nella *fig. 6* e lo esegua con il comando **RUN**.

Sullo schermo appariranno le seguenti scritte:

```
CONVERSIONE BINARIA-DECIMALE
NUMERO DECIMALE INTERO?
```

A questo punto può introdurre nel computer un numero decimale intero qualsiasi, per esempio 16; sullo schermo comparirà immediatamente il messaggio:

```
CORRISPONDE IN BINARIO: 1ØØØØ
```

Può verificare l'esattezza del valore ottenuto consultando la tabella di conversione decimale-binaria presentata nella lezione *Analisi e programmazione 9*.

2. – Forma abbreviata di alcune istruzioni

La sintassi del linguaggio BASIC consente l'impostazione abbreviata di alcune istruzioni: ciò è molto utile quando si deve battere un programma piuttosto lungo e si vogliono evitare errori formali, soprattutto per quello che riguarda le parole chiavi (che sono formulate in lingua inglese).

Come già precedentemente precisato, in quasi tutti i linguaggi BASIC, per esempio, al posto della parola chiave **PRINT** si batte il carattere ? (punto interrogativo).

Pertanto, se ad esempio si vuole calcolare in modo immediato l'espressione matematica seguente:

$$\frac{3+(1+2\emptyset)^2}{\log_e(1\emptyset)+e^2}$$

si imposterà da tastiera:

```
? (3+(1+2Ø)↑2)/(LOG(1Ø)+EXP(2))
```

seguito dal tasto **RETURN** .

Ogni sistema personal computer ha un suo proprio criterio per abbreviare le parole chiavi delle istruzioni e dei comandi BASIC.

I computer C64 e VIC-20 utilizzano il criterio esposto nella *fig. 7*, dove in corrispondenza delle parole chiavi sono riportate le operazioni di abbreviazione ed i caratteri simbolici che appaiono sul video.

Come può rilevare, le abbreviazioni (ad eccezione della parola **PRINT**) sono ottenute nel seguente modo: si battono la prima o le prime due lettere della parola chiave, quindi, tenendo premuto il tasto **SHIFT** , si batte la lettera successiva.

Per esempio, la parola chiave **AND** viene scritta in forma abbreviata così:

A **SHIFT** → **N**

In un programma le parole chiavi scritte in forma abbreviata saranno

PAROLA CHIAVE	ABBREVIAZIONE	CARATTERE SULLO SCHERMO	PAROLA CHIAVE	ABBREVIAZIONE	CARATTERE SULLO SCHERMO
ABS	A SHIFT → B	A	PEEK	P SHIFT → E	P
AND	A SHIFT → N	A	POKE	P SHIFT → O	P
ASC	A SHIFT → S	A	PRINT	?	?
ATN	A SHIFT → T	A	PRINT	P SHIFT → R	P
CHRS	C SHIFT → H	C	READ	R SHIFT → E	R
CLOSE	CL SHIFT → O	CL	RESTORE	RE SHIFT → S	RE
CLR	C SHIFT → L	C	RETURN	RE SHIFT → T	RE
CMD	C SHIFT → M	C	RIGHTS	R SHIFT → I	R
CONT	C SHIFT → O	C	RND	R SHIFT → N	R
DATA	D SHIFT → A	D	RUN	R SHIFT → U	R
DEF	D SHIFT → E	D	SAVE	S SHIFT → A	S
DIM	D SHIFT → I	D	SGN	S SHIFT → G	S
END	E SHIFT → N	E	SIN	S SHIFT → I	S
EXP	E SHIFT → X	E	SPC(	S SHIFT → P	S
FOR	F SHIFT → O	F	SQR	S SHIFT → Q	S
FRE	F SHIFT → R	F	STEP	ST SHIFT → E	ST
GET	G SHIFT → E	G	STOP	S SHIFT → T	S
GOSUB	GO SHIFT → S	GO	STRS	ST SHIFT → R	ST
GOTO	G SHIFT → O	G	SYS	S SHIFT → Y	S
INPUT	I SHIFT → N	I	TAB(	T SHIFT → A	T
LET	L SHIFT → E	L	THEN	T SHIFT → H	T
LEFTS	LE SHIFT → F	LE	USR	U SHIFT → S	U
LIST	L SHIFT → I	L	VAL	V SHIFT → A	V
LOAD	L SHIFT → O	L	VERIFY	V SHIFT → E	V
MIDS	M SHIFT → I	M	WAIT	W SHIFT → A	W
NEXT	N SHIFT → E	N			
NOT	N SHIFT → O	N			
OPEN	O SHIFT → P	O			

Fig. 7 – Abbreviazioni delle parole chiavi del linguaggio BASIC *sui computer C64 e VIC-20.*

visualizzate per esteso, quando viene dato il comando **LIST**.

Osservando la tabella della *fig. 7* può constatare che le parole chiavi **SPC** e **TAB**, quando sono scritte in forma abbreviata, comprendono anche la prima parentesi.

L'utilizzo delle parole abbreviate in un programma BASIC rende quest'ultimo meno comprensibile, a meno che non si imparino a memoria i caratteri riportati nella *fig. 7*.

Passeremo ora ad esaminare alcuni interessanti esercizi di programmazione, riguardanti le capacità grafiche dei computer C64 e VIC-20.

Come già fatto nelle precedenti lezioni, a seconda di quale dei due computer Lei è in possesso, dovrà fare riferimento al corrispondente capitolo, cioè riferirsi al capitolo 3 se dispone del C64, oppure al capitolo 4 se utilizza il VIC-20.

3. – Grafica sul computer C64

ESERCIZIO 1 – Grafica con istruzioni **READ** *e* **DATA**

Nella precedente lezione abbiamo visto come realizzare figure sullo schermo, utilizzando i caratteri grafici residenti nella memoria ROM del computer e l'istruzione **PRINT**.

L'esercizio di programmazione che ora Le viene proposto consente di visualizzare sullo schermo la figura stilizzata di un autocarro articolato, utilizzando le istruzioni **READ** e **DATA**, le quali si dimostrano particolarmente utili in programmi di grafica per richiamare i caratteri grafici predefiniti.

L'uso delle **READ** e delle **DATA** permette di ottenere figure con minore numero di istruzioni rispetto a quelle che sono necessarie per realizzare le medesime figure con l'istruzione **PRINT**.

Il listato del programma che prevede la grafica con **READ** e **DATA** è riportato nella *fig. 8*.

```
10 REM**************************
20 REM*GRAFICA CON READ E DATA*
30 REM**************************
40 PRINT"□"
50 POKE 53281,14:POKE 53280,13
60 FOR A=1 TO 5
70 FOR B=1 TO 12
80 READ X
90 POKE 1024+291+(A*40)+B,X
100 POKE 55296+291+(A*40)+B,1
110 NEXT B
120 NEXT A
130 DATA 32,102,102,102,102,102,102,102,32,32,32,32
140 DATA 32,102,2,1,19,9,3,102,32,102,77,32
150 DATA 32,102,102,102,102,102,102,102,98,102,102,32
160 DATA 32,32,81,81,32,32,32,32,32,32,81,32
170 DATA 119,119,119,119,119,119,119,119,119,119,119,119
180 END
```

Fig. 8 – Programma illustrante l'uso delle istruzioni **READ** *e* **DATA** *nella grafica del C64.*

Nell'istruzione 5Ø sono presenti due istruzioni **POKE**, che provvedono a colorare lo schermo in azzurro (codice 14) ed il margine in verde chiaro (codice 13).

Il ciclo di programma, che inizia con l'istruzione 6Ø e termina con l'istruzione 12Ø, provvede a definire le cinque istruzioni **DATA** corrispondenti alle cinque linee dello schermo occupate dal disegno dell'autocarro, il cui aspetto è riportato nella *fig. 9*.

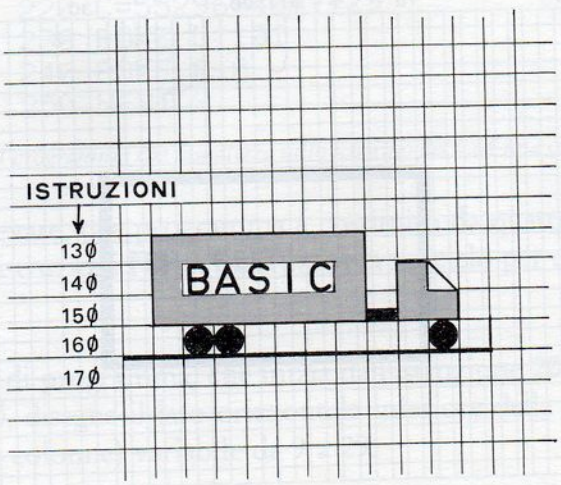


Fig. 9 – Aspetto dell'autocarro articolato riprodotto sullo schermo del C64.

I dodici valori numerici (codici grafici ROM) presenti in ciascuna istruzione **DATA** corrispondono alle dodici colonne dello schermo, definite dalla seconda procedura ciclica (istruzioni 7Ø÷11Ø), in corrispondenza delle quali vengono posizionati i caratteri grafici stessi.

I dati (caratteri grafici) contenuti nelle **DATA** sono letti dall'istruzione **READ** X (linea di programma 8Ø) e quindi depositati nelle previste locazioni di memoria caratteri del computer mediante l'istruzione **POKE**, presente nella linea di programma 9Ø.

L'istruzione 1ØØ provvede a sua volta a colorare di bianco (codice colori 1) la figura visualizzata.

Nelle istruzioni 9Ø e 1ØØ il numero 291 stabilisce il punto di partenza di esecuzione della figura, punto che corrisponde alla riga 1315 (1024+291) e alla colonna 11.

Terminata la descrizione del programma, può immetterlo nel computer ed eseguirlo con il comando **RUN**: sullo schermo comparirà la figura statica rappresentante l'autocarro (*fig. 9*).

ESERCIZIO 2 – Disegna una cornice

Vediamo ora come si deve procedere per disegnare linee orizzontali e verticali in determinate posizioni dello schermo, in modo da formare una cornice, ossia un rettangolo, come illustrato nella *fig. 10*.

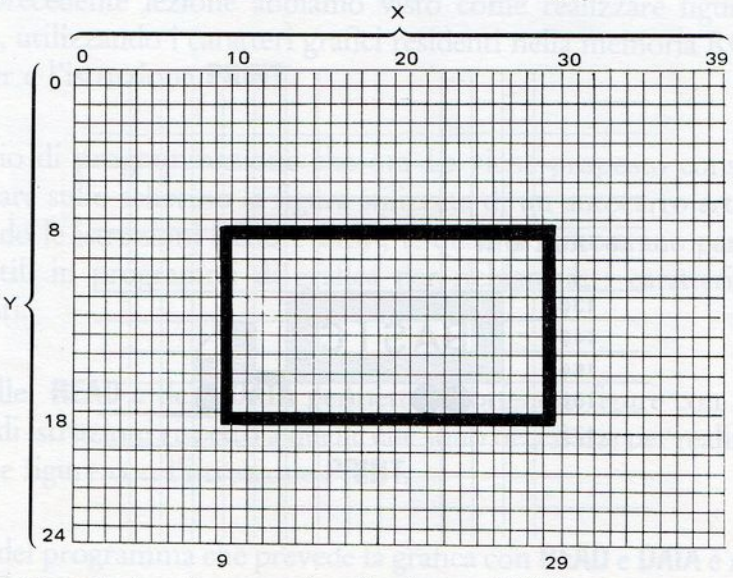


Fig. 10 – Rappresentazione di una cornice sullo schermo del C64.

In tale figura, come già precedentemente fatto, lo schermo è rappresentato suddiviso in tanti quadratini quante sono le locazioni di memoria caratteri del computer C64, allo scopo di potere individuare le coordinate, ossia i punti d'inizio e fine, di ciascuna delle quattro linee che costituiscono la cornice da disegnare.

La codifica in BASIC del programma in questione è riportata nella *fig. 11*.

```
10 REM**********************
12 REM*DISEGNA UNA CORNICE*
14 REM**********************
20 FOR X=9 TO 29
30 L=1024+40*18
35 C=55296+40*18
40 POKE L+X,100
50 POKE C+X,0
60 NEXT X
70 FOR Y=18 TO 8 STEP-1
80 L=1024+9+40*Y
90 C=55296+9+40*Y
100 POKE L,103
110 POKE C,0
120 NEXT Y
130 FOR X=10 TO 28
140 L=1024+40*7
150 C=55296+40*7
160 POKE L+X,100
170 POKE C+X,0
180 NEXT X
200 FOR Y=18 TO 8 STEP-1
210 L=1024+40*Y+29
220 C=55296+40*Y+29
230 POKE L,101
240 POKE C,0
250 NEXT Y
```

Fig. 11 – Listato del programma che visualizza sullo schermo del C64 una cornice (rettangolo).

Come può rilevare, tale programma è costituito da quattro distinti cicli realizzati con istruzioni **FOR... NEXT...**, ossia un ciclo per ciascun lato del rettangolo.

Il primo ciclo di programma, che inizia dall'istruzione 2Ø e termina con l'istruzione 6Ø, disegna il lato orizzontale inferiore della cornice con X (numero delle colonne) variabile da 9 a 29.

L'istruzione 3Ø effettua il calcolo della locazione di memoria per il

codice del carattere utilizzato per tracciare la linea. Il numero 18 presente in tale istruzione specifica la riga (orizzontale) lungo la quale verrà tracciata la linea stessa.

L'istruzione 35 calcola la corrispondente locazione di memoria colori, necessaria per "colorare" la linea disegnata con l'istruzione 3Ø.

Nell'istruzione 4Ø viene inserito (**POKE**) il codice carattere 1ØØ, costituito da un trattino orizzontale (veda a questo proposito la relativa tabella fornita nella precedente lezione pratica), nell'apposita locazione di memoria calcolata con l'istruzione 3Ø.

Analogamente, l'istruzione 5Ø provvede ad inserire il codice colore Ø (nero) nella corrispondente locazione di memoria colori calcolata con l'istruzione 35.

La lunghezza della linea orizzontale che verrà disegnata risulterà pari alla differenza fra il punto d'arrivo (colonna 29) ed il punto di partenza (colonna 9), ossia sarà uguale a 29–9 = 20 quadretti o posizioni di carattere sullo schermo.

Il secondo ciclo di programma, istruzioni 7Ø ÷ 12Ø, provvede a disegnare il lato verticale sinistro della cornice, con Y (numero delle righe) variabile da 18 a 8.

L'istruzione 8Ø calcola la locazione della memoria caratteri necessaria per tracciare la linea.

In questa istruzione il numero 9 indica la colonna in corrispondenza della quale sarà disegnata la linea verticale.

Analogamente, l'istruzione 9Ø esegue il calcolo della corrispondente locazione della memoria colori.

Le istruzioni 1ØØ e 11Ø hanno rispettivamente la funzione di inserire il carattere 1Ø3 (trattino vericale) e il colore nero (codice Ø) nelle apposite locazioni di memoria calcolate con le istruzioni 8Ø e 9Ø.

Il ciclo di programma 13Ø ÷ 18Ø disegna la linea orizzontale superiore della cornice, in corrispondenza della riga 8 con le stesse modalità adottate per il disegno della linea orizzontale inferiore.

In questo caso però X viene fatta variare da 1Ø a 28 (anziché da 9 a 29), per tenere conto dello spessore delle linee verticali.

Infine, il ciclo di programma 200÷250 disegna il lato verticale destro della cornice, con Y variabile sempre da 18 a 8.

Il numero 29 presente nelle istruzioni 210 e 220 specifica la colonna in corrispondenza della quale verrà tracciata la seconda linea verticale della cornice.

L'esame del listato è così concluso.

Dopo avere cancellato la memoria del computer con il comando **NEW** immetta nel computer stesso il nuovo programma e quindi lo esegua con il comando **RUN**: sullo schermo vedrà apparire, come previsto, un rettangolo simile a quello mostrato nella *fig. 10*.

Seguendo le istruzioni ora fornite, potrà esercitarsi a disegnare rettangoli e quadrati sullo schermo del computer, provando a cambiare i caratteri grafici utilizzati per tracciare le linee.

ESERCIZIO 3 – Programma multicolore

Questo programma consente di visualizzare a caso sullo schermo (una riga dopo l'altra, dall'alto in basso) dei quadratini colorati, ottenendo cosí un effetto grafico simile ad un mosaico, i cui colori cambiano in continuazione.

La generazione dei quadratini colorati viene ottenuta mediante due istruzioni **FOR... NEXT....**

Il listato del programma è riportato nella *fig. 12*.

```
10 REM**************************
15 REM*PROGRAMMA MULTICOLORE*
20 REM**************************
30 PRINT"J"
35 FOR X=0 TO 24
40 POKE 53281,1
45 FOR A=0 TO 15
50 B=INT(RND(1)*40)
55 POKE 1024+(40*X)+B,160
60 POKE 55296+(40*X)+B,A
65 NEXT A
70 NEXT X
75 GOTO 35
```

Fig. 12 – Codifica in linguaggio BASIC del programma che visualizza sullo schermo del C64 una serie di quadratini colorati a caso.

Il ciclo di programma costituito dalle istruzioni 35÷7Ø esegue il conteggio delle righe dello schermo sulle quali verranno visualizzati i quadratini colorati.

L'istruzione 4Ø provvede a colorare di bianco lo schermo per fare risaltare meglio i colori.

L'istruzione 45 imposta un'iterazione, che fornisce alla variabile A i valori compresi fra Ø e 15, corrispondenti al codice dei colori da inserire (**POKE**) nella locazione per la memoria del colore (istruzione 6Ø).

L'istruzione 5Ø genera un numero casuale intero compreso fra Ø e 4Ø (numero delle colonne del C64) che viene utilizzato nelle istruzioni 55 e 6Ø per visualizzare a caso, su ciascuna riga dello schermo, il numero dei quadratini colorati.

Questi ultimi sono ottenuti mediante il numero di codice 16Ø, che corrisponde al carattere 32 (spazio) in negativo: infatti, 128+32=16Ø.

Infine, l'istruzione 75 provvede a rimandare il programma all'istruzione 35, da dove l'elaborazione riprende.

Cancelli il precedente programma ed immetta quello nuovo ora descritto nel computer, quindi lo esegua con il comando **RUN**: verrà visualizzata una successione di quadratini colorati a caso, che si disporranno in modo imprevisto su ciascuna riga dello schermo.

Per fermare l'elaborazione e fare ritornare il video nelle sue condizioni standard (schermo blu e bordo azzurro) prema il tasto **RUN | STOP** e, mantendendolo abbassato, batta il tasto **RESTORE**.

3.1 – Grafica con gli sprite

Con il termine **sprite** (che in inglese significa "folletto" o "spiritello") viene indicata una figura ad alta risoluzione, che può essere creata e movimentata sullo schermo mediante apposite istruzioni BASIC.

Con gli sprite possono essere generate figure con qualsiasi profilo, ottenendo cosí effetti di animazione assai migliori di quelli che si possono avere utilizzando i simboli grafici residenti nella memoria ROM.

Gli sprite sono controllati da un particolare circuito per la gestione video

incorporato nel C64 (denominato 6567 oppure VICII), il quale permette di realizzare sino ad un massimo di otto sprite contemporaneamente.

Ogni sprite occupa un'area di 24×21 pixel.

Ricordi che i pixel (picture cell) sono i punti grafici o elementi che compongono lo schermo grafico del televisore, il quale, nella condizione ad alta risoluzione, risulta formato da un'area larga 320 punti ed alta 200 punti.

Essendo lo sprite costituito da 24×21=504 punti, occorrono 504 bit d'informazione per rappresentarlo, ossia **63 byte organizzati in 21 righe di 3 byte ciascuna**, come chiaramente indicato nella *fig. 13*.

Per programmare uno sprite è necessario definire la sequenza dei byte che lo costituiscono, scegliere il colore con il quale si desidera rappresentarlo, stabilire la sua posizione sullo schermo, renderlo visibile, ossia abilitarlo.

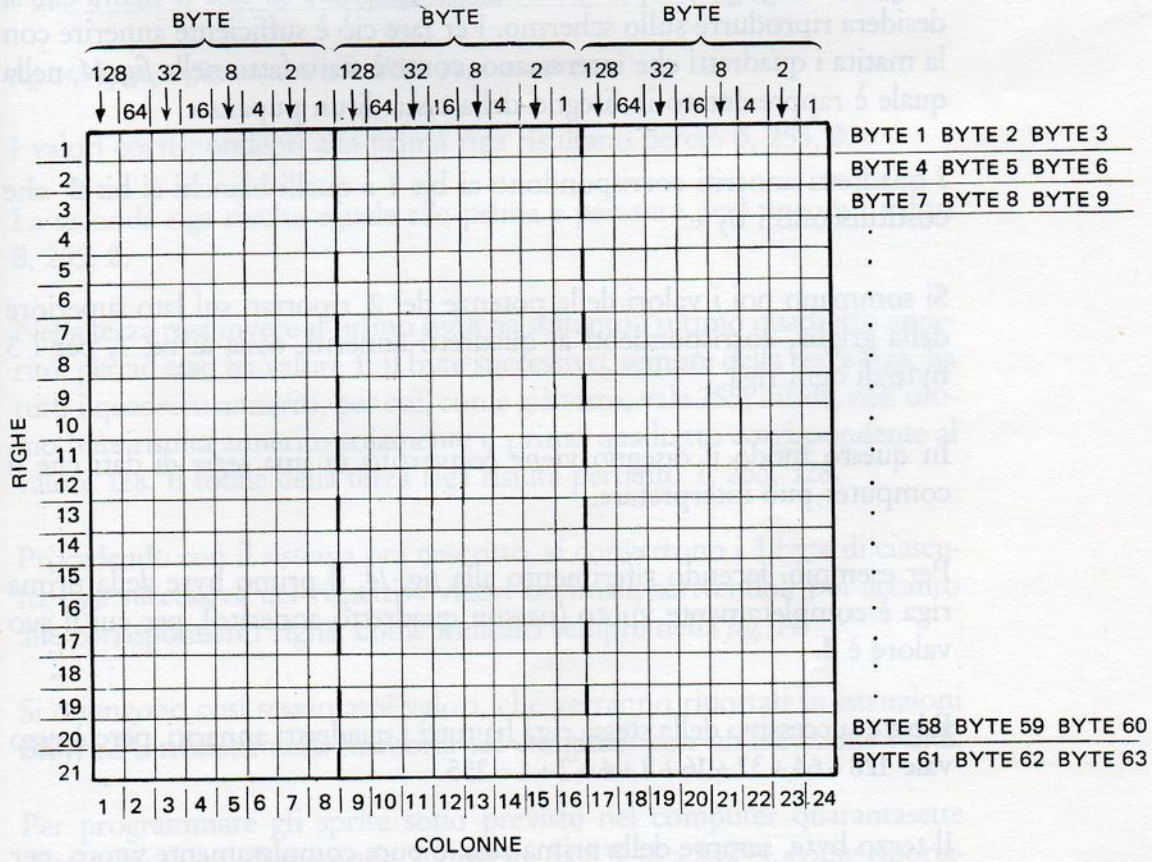


Fig. 13 – Griglia per la definizione di uno sprite sullo schermo del C64.

Vediamo ora come queste fasi vengono praticamente eseguite.

Innanzitutto è necessario disegnare su un foglio di carta quadrettata il perimetro di un rettangolo, avente la base di 24 quadretti e l'altezza di 21 quadretti, come si vede nella *fig. 13*.

Si divide poi verticalmente il rettangolo cosí disegnato in tre parti, ognuna delle quali costituita da otto quadretti: ciascuna serie di otto quadretti rappresenta 1 byte.

Sul lato superiore del rettangolo, che denomineremo **griglia**, si scrivono poi, in corrispondenza di ciascuna colonna di quadretti, le potenze del 2, ossia i numeri 128, 64, 32, 16, 8, 4, 2, 1, i quali si ripetono tre volte (cioè una volta per ciascuno dei 3 byte che costituiscono le 21 righe dello sprite).

Si riporta, infine, in corrispondenza del lato sinistro del rettangolo, la numerazione delle righe da 1 a 21.

Preparata la griglia, si può procedere a disegnare su essa la figura che si desidera riprodurre sullo schermo. Per fare ciò è sufficiente annerire con la matita i quadretti che interessano, come è stato fatto nella *fig. 14*, nella quale è rappresentato il disegno della testa di un pupazzo.

I quadretti anneriti corrispondono ai bit 1 e quelli bianchi ai bit Ø che costituiscono i byte.

Si sommano poi i valori delle potenze del 2, riportati sul lato superiore della griglia, corrispondenti ai quadretti anneriti, ossia ai bit 1, per i 3 byte di ogni riga.

In questo modo il disegno viene convertito in una serie di dati che il computer può interpretare.

Per esempio, facendo riferimento alla *fig. 14*, il primo byte della prima riga è completamente vuoto (nessun quadretto annerito), per cui il suo valore è Ø.

Il byte successivo della stessa riga ha tutti i quadretti anneriti, perciò esso vale 128+64+32+16+8+4+2+1=255.

Il terzo byte, sempre della prima riga, è pure completamente vuoto, per cui il suo valore è Ø.

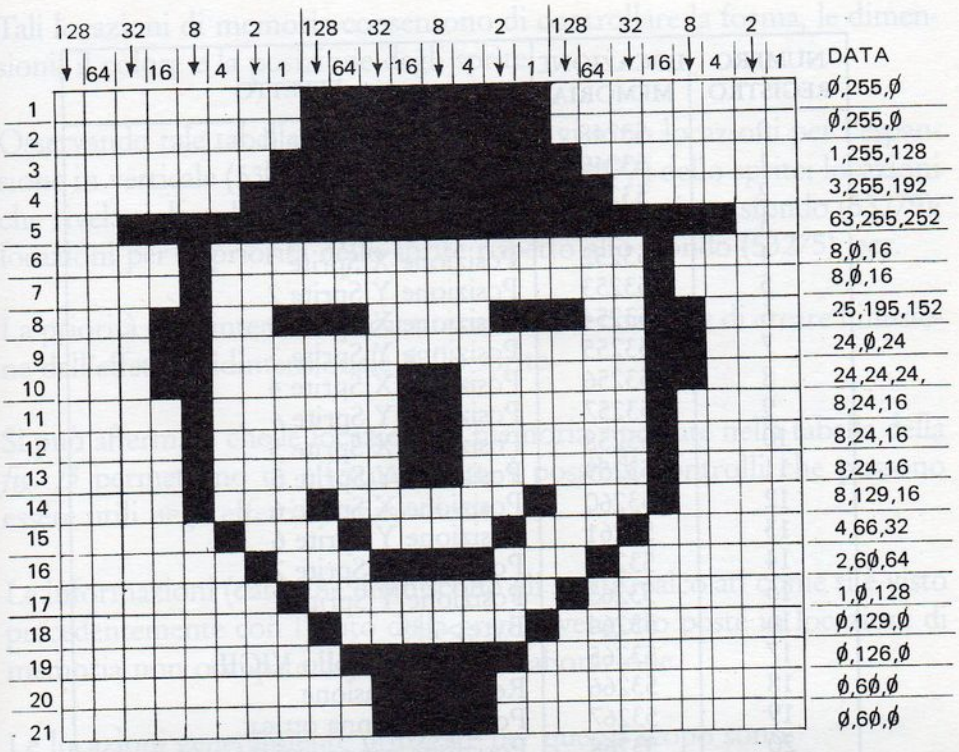


Fig. 14 – Sprite raffigurante la testa di un pupazzo sul C64.

I valori corrispondenti alla prima riga risultano perciò Ø, 255, Ø.

La seconda riga risulta eguale alla prima e pertanto avrà ancora i valori Ø, 255, Ø.

Nella terza riga invece il primo byte ha soltanto l'ultimo quadretto annerito, perciò esso ha valore 1; il byte successivo, sempre della terza riga, ha tutti i quadretti anneriti, per cui, come già visto, vale 255; infine, nell'ultimo byte risulta annerito solamente il primo quadretto corrispondente al valore 128. Il totale della terza riga risulta pertanto 1, 255, 128.

Procedendo con il sistema ora descritto, si convertono i 3 byte di ciascuna riga successiva nei rispettivi valori decimali, scrivendoli poi accanto alle corrispondenti righe, come indicato sempre nella *fig. 14*.

Si ottengono cosí sessantatré valori, che verranno riportati in istruzioni **DATA** ed introdotti nella memoria del computer con un'istruzione **POKE**.

Per programmare gli sprite sono previste nel computer quarantasette diverse locazioni di memoria, numerate da **53248** a **53294**, come riportato nella tabella della *fig. 15*.

NUMERO REGISTRO	LOCAZIONE MEMORIA	EFFETTO
0	53248	Posizione X Sprite Ø
1	53249	Posizione Y Sprite Ø
2	53250	Posizione X Sprite 1
3	53251	Posizione Y Sprite 1
4	53252	Posizione X Sprite 2
5	53253	Posizione Y Sprite 2
6	53254	Posizione X Sprite 3
7	53255	Posizione Y Sprite 3
8	53256	Posizione X Sprite 4
9	53257	Posizione Y Sprite 4
10	53258	Posizione X Sprite 5
11	53259	Posizione Y Sprite 5
12	53260	Posizione X Sprite 6
13	53261	Posizione Y Sprite 6
14	53262	Posizione X Sprite 7
15	53263	Posizione Y Sprite 7
16	53264	Byte > 255
17	53265	Registro controllo VICII
18	53266	Registro scansione
19	53267	Posizione penna ottica
20	53268	Posizione penna ottica
21	53269	Abilitazione sprite
22	53270	Registro controllo VICII
23	53271	Espansione sprite in verticale (Y)
24	53272	Controllo memoria VICII
25	53273	Registro delle interruzioni
26	53274	Maschera per le interruzioni
27	53275	Priorità sullo sfondo
28	53276	Selezione multicolore
29	53277	Espansione sprite in orizzontale (X)
30	53278	Collisione fra sprite
31	53279	Collisione sprite-sfondo
32	53280	Colore margine schermo
33	53281	Colore sfondo Ø
34	53282	Colore sfondo 1
35	53283	Colore sfondo 2
36	53284	Colore sfondo 3
37	53285	Sprite multicolore 1
38	53286	Sprite multicolore 2
39	53287	Colore Sprite Ø
40	53288	Colore Sprite 1
41	53289	Colore Sprite 2
42	53290	Colore Sprite 3
43	53291	Colore Sprite 4
44	53292	Colore Sprite 5
45	53293	Colore Sprite 6
46	53294	Colore Sprite 7

Fig. 15 – Locazioni di memoria per l'uso degli sprite nel C64.

Pagina mancante

Pagina mancante

coordinate X, Y, ossia la posizione orizzontale e quella verticale dello schermo in cui verrà visualizzato lo sprite Ø.

La posizione orizzontale dello sprite è data dalla locazione V(=53248)+Ø (valore corrispondente alla posizione X dello sprite Ø, come indicato dalla tabella della *fig. 15*).

La posizione verticale dello sprite è invece fornita dalla locazione V(=53248)+1, valore corrispondente alla posizione Y dello sprite Ø.

I valori delle coordinate X iniziano dal lato sinistro dello schermo e vanno da Ø a 32Ø, mentre quelli delle coordinate Y iniziano dal lato superiore dello schermo e vanno da Ø a 2ØØ.

Nelle istruzioni riportate nella linea di programma 65 i valori scelti per le coordinate X e Y sono rispettivamente 17Ø e 135, corrispondenti alla visualizzazione dello sprite all'incirca al centro dello schermo.

L'istruzione 7Ø provvede ad abilitare lo sprite, ossia a dare il consenso per la sua visualizzazione. Infatti, osservando la tabella della *fig. 15* si rileva che il numero 21 corrisponde appunto al numero di registro relativo a tale funzione, ossia alla locazione di memoria 53269, che è eguale a V(53248)+21.

Per fare in modo che lo sprite appaia sullo schermo, occorre infine assegnargli il valore 1 (e per farlo scomparire il valore Ø).

Le istruzioni **DATA** comprese fra le linee di programma 1ØØ e 16Ø contengono tutti i dati relativi alla formazione della figura.

Concluso l'esame del programma della *fig. 16*, può immetterlo nel computer battendo sulla tastiera le istruzioni che lo costituiscono.

Verifichi poi, come al solito, che non vi siano errori, quindi lo esegua battendo il comando **RUN**: a circa metà schermo apparirà la testa del pupazzo della *fig. 14* (avente le dimensioni di 3×2,5 cm circa) nel colore bianco.

Volendo rappresentare lo sprite con altri colori, occorre fornire un'apposita istruzione.

Per fare ciò è necessario assegnare il numero corrispondente al codice del colore prescelto (che, come è noto, è compreso fra Ø e 15) all'apposita

Pagine mancanti

PARLA BASIC

Corso teorico-pratico per corrispondenza di

ANALISI E PROGRAMMAZIONE BASIC

Scuola Radio Elettra

10126 Torino - Via Stellone 5

Tel. (011) 674432